Manuel de rèfèrence MySQL

Table des matières

1 Informations gènèrales

Le logiciel MySQL (TM) est un serveur de base de donnèes SQL trés rapide, multi-threadè, multi-utilisateur et robuste. Le serveur MySQL est destinè aux missions stratègiques et aux systémes de production á forte charge, ainsi qu'á l'intègration dans des logiciels dèployès á grande èchelle. MySQL est une marque dèposèe de MySQL AB.

Le logiciel MySQL dispose de deux licenses. Les utilisateurs peuvent choisir entre utiliser MySQL comme un logiciel Open Source/Logiciel libre, sous les termes de la licence GNU General Public License (http://www.gnu.org/licenses/) ou bien, ils peuvent acheter une licence commerciale auprés de MySQL AB. Voir Section 1.4 [Licensing and Support], page 16.

Le site web de MySQL (http://www.mysql.com/) fournit les derniéres informations sur le serveur MySQL.

La liste suivante dècrit les sections particuliéres de ce manuel :

- Pour plus d'informations sur la sociètè derriére le serveur de base de donnèes MySQL, voyez Section 1.3 [What is MySQL AB], page 12.
- Pour une prèsentation des capacitès de serveur de base de donnèes MySQL, voyez Section 1.2.2 [Features], page 5.
- Pour les instructions d'installation, voyez Chapitre 2 [Installing], page 69.
- Pour des conseils sur le port du serveur de base de donnèes MySQL sur de nouvelles architectures ou systémes d'exploitation, voyez Annexe E [Porting], page 814.
- Pour des informations sur la mise á jour vers la version 3.23, voyez Section 2.5.1 [Upgrading-from-3.23], page 106.
- Pour des informations sur la mise á jour vers la version 3.22, voyez Section 2.5.2 [Upgrading-from-3.22], page 109.
- Pour une introduction au serveur de base de donnèes MySQL, voyez Chapitre 3 [Tutorial], page 153.
- Pour des exemples de SQL et des tests de performances, voyez le dossier de tests ('sql-bench' de la distribution.
- Pour connaître l'historique des fonctionnalitès et bogues, voyez Annexe D [News], page 727.
- Pour une liste des bogues connus et des limitations, voyez Section 1.7.5 [Bugs], page 43.
- Pour les plans de dèveloppement, voyez Section 1.8 [TODO], page 46.
- Pour une liste de tous les contributeurs á ce projet, voyez Annexe C [Credits], page 717.

Important:

Les rapports d'erreurs (aussi appelès bogues), ainsi que les questions et commentaires, doivent être envoyès á la liste de diffusion mysql@lists.mysql.com. Voir Section 1.6.2.3 [Bug reports], page 27.

Le script mysqlbug doit être utilisè pour gènèrer le rapport de bogues. Pour les distributions sources, le script mysqlbug est accessible dans le dossier 'scripts'. Pour les distributions binaires, mysqlbug est installè dans le dossier 'bin'. Si vous avez trouvè un probléme de sècuritè critique dans le code du serveur MySQL, vous devez envoyez un email á security@mysql.com.

1.1 A propos du manuel

Ceci est le manuel de rèfèrence de MySQL; il documente MySQL jusqu'á la version 4.1.1-alpha. Les èvolutions fonctionnelles sont toujours indiquèes avec une rèfèrence á la version d'èvolution, de maniére á ce que ce manuel soit toujours valable, même si vous utilisez une ancienne version de MySQL. Etant un manuel de rèfèrence, il ne fournit aucune description gènèrale sur le langage SQL ou les concepts de bases de donnèes relationnelles.

Comme le logiciel de base de donnèes MySQL est en dèveloppement constant, ce manuel es mis á jour frèquemment. La version la plus rècente est disponibles á http://www.mysql.com/documentation/ en diffèrents formats, incluant HTML, PDF et Windows HLP.

L'original du document est un fichier au format Texinfo. La version HTML est produite automatiquement avec une version modifièe de texi2html. La version en texte plein et version Info sont produites par makeinfo. La version PostScript est produite avec texi2dvi et dvips. La version PDF est produite avec pdftex.

Si vous avez du mal á trouver des informations dans ce manuel, vous pouvez essayer notre version avec moteur de recherche, sur notre site web : http://www.mysql.com/doc/.

Si vous avez des suggestions concernant des ajouts ou des corrections á ce manuel, vous pouvez les envoyez á l'èquipe de documentation á docs@mysql.com.

Ce manuel a ètè ècrit initialement par David Axmark et Michael (Monty) Widenius. Il est actuellement entretenu par Michael (Monty) Widenius, Arjen Lentz et Paul DuBois. Pour les autres contributeurs, voyez les Annexe C [Credits], page 717.

La traduction de ce manuel a ètè faite sous la direction de Damien Seguy, et organisèe par Mehdi Achour. David Manusset, Guillaume Plessis, Patrick Haond, Sylvain Maugiron et Yannick Torres ont contribuè largement á cette traduction et son entretien.

1.1.1 Conventions utilisèes dans ce manuel

Ce manuel utilise certaines conventions typographiques :

`constant` La police á largeur fixe est utilisèe pour les noms de commandes et les options, les requêtes SQL, les noms de bases de donnèes, de tables et de colonnes, le code C et Perl, les variables d'environnement. Par exemple, "Pour voir comment `mysqladmin` fonctionne, exècutez-le avec l'option `--help`."

'`filename`' La police á largeur fixe avec des guillemets d'encadrement indique des noms de fichiers et de chemins de dossiers. Par exemple : "La distribution est installèe dans le dossier '`/usr/local/`'."

'`c`' La police á largeur fixe avec des guillemets d'encadrement est aussi utilisèe pour indiquer des sèquences de caractéres. Par exemple : "Pour spècifier un caractére joker, utilisez le caractére '`%`'."

italic Les polices en italique sont utilisèes pour attirer l'attention, *comme ceci*.

boldface Le gras est utilisè pour les entêtes de tables, et aussi pour **attirer fortement votre attention**.

Lorsque les commandes qui sont affichèes sont destinèes á être exècutèes par un programme particulier, le nom du programme est indiquè dans l'invite de la commande. Par exemple, `shell>` indique une commande que vous exècutez depuis votre console shell, et `mysql>` indique une commande que vous exècutez depuis le client `mysql` :

```
shell> tapez une commande shell ici
mysql> tapez une requête SQL ici
```

Les commandes shell sont affichèes en utilisant la syntaxe du Bourne. Si vous utilisez le style `csh`-shell, vous aurez peut être a adapter lègérement les commandes. Par exemple, la syntaxe de modification d'une variable et d'exècution d'une commande ressemble á ceci en Bourne shell :

```
shell> NOMVAR=valeur une_commande
```

En `csh`, vous auriez a exècuter la commande suivante :

```
shell> setenv NOMVAR valeur
shell> une_commande
```

Souvent, les noms de bases de donnèes, tables ou colonnes doivent être remplacès dans les commandes. Pour indiquer qu'une telle substitution est nècessaire, ce manuel utilise les noms de `nom_de_base`, `nom_de_table` et `nom_colonne`. Par exemple, vous pourriez avoir une requête comme ceci :

```
mysql> SELECT nom_colonne FROM nom_de_base.nom_de_table;
```

Cela signifie que si vous devez saisir une requête semblable, vous devriez utiliser votre propre nom de colonne, table et base de donnèes, ce qui pourrait se traduire par ceci :

```
mysql> SELECT author_name FROM biblio_db.author_list;
```

Les mot rèservès SQL ne sont pas sensibles á la casse, et peuvent être ècrits en majuscules ou minuscules. Ce manuel utilise les majuscules.

Dans les illustrations de syntaxe, les crochets (‘`[`’ et ‘`]`’) sont utilisès pour indiquer des clauses ou mots optionnels. Par exemple, dans la requête suivante, `IF EXISTS` est optionnel :

```
DROP TABLE [IF EXISTS] nom_de_table
```

Lorsqu'un èlèment de syntaxe est constituè d'un certain nombre d'alternatives, les alternatives sont sèparèes par des barres verticales (‘`|`’). Lorsqu'un membre d'un tel jeu de possibilitès **peut** être choisi, les alternatives sont listèes entre crochets (‘`[`’ et ‘`]`’):

```
TRIM([[BOTH | LEADING | TRAILING] [remstr] FROM] str)
```

Lorsqu'un èlèment d'un jeu de possibilitès **doit** être choisi, les alternatives sont placèes entre accolades (‘`{`’ et ‘`}`’):

```
{DESCRIBE | DESC} nom_de_table {nom_colonne | wild}
```

1.2 Qu'est ce que MySQL?

`MySQL`, le plus populaire des serveurs de bases de donnèes SQL `Open Source`, est dèveloppè, distribuè et supportè par `MySQL AB`. `MySQL AB` est une sociètè commerciale, fondèe par les

dèveloppeurs de MySQL, qui dèveloppent leur activitè en fournissant des services autour de MySQL. Voir Section 1.3 [What is MySQL AB], page 12.

Le site web de MySQL (http://www.mysql.com/) fournit les toutes derniéres actualitès sur le logiciel MySQL et sur la sociètè MySQL AB.

MySQL est un systéme de gestion de bases de donnèes.

Une base de donnèes est un ensemble organisè de donnèes. Cela peut aller d'une simple liste de courses au supermarchè á une galerie de photos, ou encore les grands systémes d'informations des multi-nationales. Pour ajouter, lire et traiter des donnèes dans une base de donnèes, vous avez besoin d'un systéme de gestion de bases de donnèes tel que le serveur MySQL. Comme les ordinateurs sont trés bons á manipuler de grandes quantitès de donnèes, le systéme de gestion de bases de donnèes joue un rôle central en informatique, aussi bien en tant qu'application á part entiére, qu'intègrè dans d'autres logiciels.

MySQL est un serveur de bases de donnèes relationnel.

Un serveur de bases de donnèes stocke les donnèes dans des tables sèparèes plutôt que de tout rassembler dans une seule table. Cela amèliore la rapiditè et la souplesse de l'ensemble. Les tables sont reliées par des relations dèfinies, qui rendent possible la combinaison de donnèes entre plusieurs tables durant une requête. Le SQL dans "MySQL" signifie "Structured Query Language" : le langage standard pour les traitements de bases de donnèes.

MySQL est Open Source.

Open Source (Standard Ouvert) signifie qu'il est possible á chacun d'utiliser et de modifier le logiciel. Tout le monde peut tèlècharger MySQL sur Internet, et l'utiliser sans payer aucun droit. Toute personne en ayant la volontè peut ètudier et modifier le code source pour l'adapter á ses besoins propres. Le logiciel MySQL utilise la licence GPL (GNU General Public License), http://www.gnu.org/licenses/, pour dèfinir ce que vous pouvez et ne pouvez pas faire avec ce logiciel, dans diffèrentes situations. Si vous ne vous sentez pas confortable avec la licence GPL ou bien que vous devez intègrer MySQL dans une application commerciale, vous pouvez acheter une licence commercial auprés de MySQL AB. Voir Section 1.4.3 [MySQL licenses], page 17.

Pourquoi utiliser le serveur de bases de donnèes MySQL?

Le serveur de bases de donnèes MySQL est trés rapide, fiable et facile á utiliser. Si c'est ce que vous recherchez, vous devriez faire un essai. Le serveur de bases de donnèes MySQL dispose aussi de fonctionnalitès pratiques, dèveloppèes en coopèration avec nos utilisateurs. Vous pouvez trouver une comparaison des performances du serveur MySQL avec d'autres systémes de bases de donnèes dans nos pages de tests de performances. Voir Section 5.1.4 [MySQL Benchmarks], page 360.

Le serveur MySQL a ètè dèveloppè á l'origine pour gèrer de grandes bases de donnèes plus rapidement que les solutions existantes, et a ètè utilisè avec succés dans des environnements de production trés contraints et trés exigeants, depuis plusieurs annèes. Bien que toujours en dèveloppement, le Le serveur MySQL offre des fonctions nombreuses et puissantes. Ses possibilitès de connexions, sa

rapiditè et sa sècuritè font du `serveur MySQL` une serveur hautement adaptè á Internet.

Les caractèristiques techniques du `serveur MySQL`.

Pour des dètails techniques avancès, voyez Chapitre 6 [Reference], page 404. Le `logiciel de bases de donnèes MySQL` est un systéme client/serveur, constituè d'un serveur `SQL` multi-threadè qui supporte diffèrents systémes de stockage, plusieurs logiciels clients et librairies, outils d'administration, ainsi que de nombreuses interfaces de programmation (des `API`).

Nous fournissons aussi le `serveur MySQL` sous la forme d'une librairie multi-threadè que vous pouvez inclure dans vos applications, pour obtenir un produit plus compact, plus rapide et plus facile á gèrer.

Il existe un grand nombre de contributions á `MySQL`.

Il est trés probable que vous pourrez trouver votre èditeur prèfèrè ou que votre environnement de programmation supporte dèjá le `serveur de base de donnèes MySQL`.

La prononciation officielle de `MySQL` est “My Ess Que Ell” (en anglais), ce qui donne “Maille Esse Cu Elle” en phonètique française. Evitez d'utiliser la prononciation “my sequel”, mais nous ne nous formaliserons pas que vous utilisiez “my sequel” ou une autre prononciation adaptèe.

1.2.1 Histoire de MySQL

Nous avons dèbutè avec l'intention d'utiliser `mSQL` pour se connecter á nos tables en utilisant nos propres routines bas niveau ISAM. Cependant, aprés quelques tests, nous sommes arrivès á la conclusion que `mSQL` n'ètait pas assez rapide et flexible pour nos besoins. Cela nous a conduit á crèer une nouvelle interface SQL pour notre base de donnèes, mais en gardant la même API que `mSQL`. Cette API a ètè choisie pour la facilitè de port des programmes de tiers.

Les liens avec le nom `MySQL` ne sont pas parfaitement ètablis. Notre dossier de base et un grand nombre de librairies et outils ètaient prèfixès par “my” depuis plus de 10 ans. Mais la fille de Monty, plus jeune que lui, ètait aussi appelèe My. Lequel des deux a conduit au nom de `MySQL` est toujours un mystére, même pour nous.

1.2.2 Les fonctionnalitès principales de MySQL

La liste suivante dècrit les caractèristiques principales du `logiciel de bases de donnèes MySQL`. Voir Section 1.5 [MySQL 4.0 In A Nutshell], page 21.

Interne et portabilitè

- Ecrit en C et C++. Testè sur un large èventail de compilateurs diffèrents.
- Fonctionne sur de nombreuses plate-formes. Voir Section 2.2.3 [Which OS], page 74.
- Utilise GNU Automake, Autoconf et Libtool pour une meilleure portabilitè.
- Dispose d'API pour C, C++, Eiffel, Java, Perl, PHP, Python, Ruby et Tcl. Voir Chapitre 8 [Clients], page 591.

- Complétement multi-threadè, grâce aux threads du noyau. Cela signifie que vous pouvez l'utiliser facilement sur un serveur avec plusieurs processeurs.
- Tables B-tree trés rapide, avec compression d'index.
- Systéme l'allocation mèmoire trés rapide, exploitant les threads.
- Jointures trés rapides, exploitant un systéme de jointures multiples en une seule passe optimisè.
- Tables en mèmoire, pour rèaliser des tables temporaires.
- Les fonctions SQL sont implèmentèes grâce á une librairie de classes optimisèes, qui sont aussi rapides que possible! Gènèralement, il n'y a aucune allocation mèmoire une fois que la requête a ètè initialisèe.
- Le code de `MySQL` est vèrifiè avec Purify (un utilitaire de dètection des fuites mèmoires commercial) ainsi qu'avec Valgrind, un outil GPL (`http://developer.kde.org/~sewardj/`).

Column Types

- Nombreux types de colonnes : entiers signès ou non, de 1, 2, 3, 4, et 8 octets, `FLOAT`, `DOUBLE`, `CHAR`, `VARCHAR`, `TEXT`, `BLOB`, `DATE`, `TIME`, `DATETIME`, `TIMESTAMP`, `YEAR`, `SET` et `ENUM`. Voir Section 6.2 [Column types], page 416.
- Enregistrements de taille fixe ou variable.
- Toutes les colonnes ont des valeurs par dèfaut. Vous pouvez utiliser la commande `INSERT` pour insèrer un sous ensemble de colonnes : les colonnes qui ne sont pas explicitement citès prennent alors leur valeur par dèfaut.

Commandes et fonctions

- Support complet des opèrateurs et fonctions dans la commande `SELECT` et la clause `WHERE`. Par exemple :

```
mysql> SELECT CONCAT(first_name, " ", last_name)
    -> FROM tbl_name
    -> WHERE income/dependents > 10000 AND age > 30;
```

- Support complet des clauses SQL `GROUP BY` et `ORDER BY`. Support des fonctions de groupages (`COUNT()`, `COUNT(DISTINCT ...)`, `AVG()`, `STD()`, `SUM()`, `MAX()` et `MIN()`).
- Support des clauses `LEFT OUTER JOIN` et `RIGHT OUTER JOIN` avec les syntaxes ANSI SQL et ODBC.
- Les alias de tables et colonnes sont compatibles avec le standard SQL92.
- `DELETE`, `INSERT`, `REPLACE` et `UPDATE` retourne le nombre de lignes affectèes. Il est possible d'obtenir le nombre de lignes trouvèes en modifiant une option lors de la connexion au serveur.
- La commande spècifique á `MySQL SHOW` est utilisèe pour obtenir des informations sur les bases, tables et index. La commande `EXPLAIN` sert á optimiser les requêtes.
- Les noms de fonctions ne sont jamis en conflit avec les noms de tables ou colonnes. Par exemple, `ABS` est un nom de colonne valide. La seule restriction est que, lors d'un appel de fonction, aucun espace n'est tolèrè

entre le nom de la fonction et la parenthése ouvrante '(' suivante. Voir Section 6.1.7 [Reserved words], page 414.

- Vous pouvez utiliser simultanèment des tables de diffèrentes bases (depuis la version 3.22).

Securitè

- Un systéme de droits et de mots de passe trés souple et sècuritaire, qui vèrifie aussi les hôtes se connectant. Les mots de passe sont bien protègès, car tout les èchanges de mot de passe sont chiffrès, même lors des connexions.

Charges supportèes et limites

- Gére les trés grandes bases de donnèes. Nous utilisons le `serveur MySQL` avec des bases qui contiennent 50 millions de lignes et nous connaissons des utilisateurs qui utilisent le `serveur MySQL` avec plus de 60 000 tables and et 5 000 000 000 (milliards) de lignes.
- Jusqu'á 32 index sont permis par table. Chaque index est constituè de 1 á 16 colonnes ou parties de colonnes. La taille maximale d'un index est de 500 octets (ce qui peut être configurè á la compilation du `serveur MySQL`. Un index peut utiliser un prèfixe issu d'un champs `CHAR` ou `VARCHAR`.

Connectivitè

- Les clients peuvent se connecter au serveur `MySQL` en utilisant les sockets TCP/IP, les sockets Unix (Unix) ou les Named Pipes (NT).
- Support de `ODBC` (Open-DataBase-Connectivity) pour Win32 (avec les sources). Toutes les fonctions ODBC 2.5 et de nombreuses autres. Par exemple, vous pouvez utiliser MS Access pour vous connecter au serveur `MySQL`. Voir Section 8.3 [ODBC], page 599.

Traductions

- Le serveur fournit des messages d'erreurs au client dans de nombreuses langues, y compris le français. Voir Section 4.6.2 [Languages], page 287.
- Support complet de plusieurs jeux de caractéres, comprenant ISO-8859-1 (Latin1), german, big5, ujis, and more. Par exemple, les caractéres nordiques 'Â', 'ä' et 'ö' sont autorisès dans les noms de tables et colonnes.
- Toutes les donnèes sont sauvèes dans le jeu de caractéres choisi. Les comparaisons normales de chaînes sont insensibles á la casse.
- Le tri est fait en fonction du jeu de caractéres choisi (par dèfaut, le jeu suèdois). Il est possible de le changer lorsque le serveur `MySQL` est dèmarrè. Pour voir un exemple trés avancè de tri, voyez le code de tri pour le Tchéque. Le `serveur MySQL` supporte de nombreux jeux de caractéres qui peuvent être spècifiè á la compilation et durant l'exècution.

Clients et utilitaires

- Inclut `myisamchk`, un utilitaire rapide pour vèrifier les tables, les optimiser et les rèparer. Toutes les fonctionnalitès de `myisamchk` sont aussi disponibles via l'interface SQL. Voir Chapitre 4 [MySQL Database Administration], page 192.

- Tous les programmes `MySQL` peuvent être appelès avec l'option `--help` ou `-?` pour obtenir de l'aide en ligne.

1.2.3 Jusqu'á quel point MySQL est il stable ?

Cette section rèpond aux questions "*Jusqu'á quel point MySQL est il stable ?*" et "*Puis-je faire confiance á MySQL pour mon projet ?*" Nous allons tenter d'apporter des rèponses claires á ces questions importantes qui concernent tous les utilisateurs potentiels. Les informations de cette section sont fournies par les listes de diffusions, qui sont trés actives et promptes á identifier les problémes et les rapporter.

Le code original date du dèbut des annèes 80 et fournit une base de code stable, tout en assurant une compatibilitè ascendante avec le format ISAM. A TcX, le prèdècesseur de `MySQL AB`, le code de `MySQL` a fonctionnè sur des projets depuis la mi 1996, sans aucun probléme. Lorsque le `Serveur MySQL` a ètè livrè á un public plus large, nous avons rèalisè qu'il contenait du code "jamais testè" qui a ètè rapidement identifiè par les utilisateurs, qui effectuait des requêtes diffèrentes des nôtres. Chaque nouvelle version avait moins de problémes de portabilitè, même si chaque nouvelle version avait de nombreuses nouvelles fonctionnalitès.

Chaque version du `Serveur MySQL` ètait parfaitement fonctionnelle. Les seuls problémes ètaient rencontrès par les utilisateurs de code de ces "zone d'ombres". Naturellement, les nouveaux utilisateurs ne connaissent pas ces zones : cette section tente de les prèsenter, dans la mesure de nos connaissances. Les descriptions correspondent surtout aux versions 3.23 du `Serveur MySQL`. Tous les bogues connus et rapportès ont ètè corrigès dans la derniére version, á l'exception de ceux qui sont listès dans la section Bugs, qui sont des problémes de conception. Voir Section 1.7.5 [Bugs], page 43.

La conception du `serveur MySQL` est faite en plusieurs couches, avec des modules indèpendants. Certains des modules les plus rècents sont listès ici, avec leur niveau de test :

Rèplication – Gamma
: De grands serveurs en grappe utilisant la rèplication sont en production, avec de bons rèsultats. L'amèlioration de la rèplication continue avec `MySQL` 4.x.

Tables `InnoDB` – Stable (en 3.23 depuis 3.23.49)
: Le gestionnaire transactionnel de tables `InnoDB` a ètè dèclarè stable en `MySQL` version 3.23, á partir de la version 3.23.49. `InnoDB` est utilisè dans de grands systémes complexes, avec forte charge.

Tables `BDB` – Gamma
: Le code de `Berkeley DB` est trés stable, mais nous sommes encore en train d'amèliorer l'interface du gestionnaire transactionnel de table `BDB` du `serveur MySQL`. Cela demande encore du temps pour qu'il soit aussi bien testè que les autres types de tables.

`FULLTEXT` – **Beta**
: La recherche en texte plein fonctionne mais n'est pas encore largement adoptèe. Des amèliorations importantes sont prèvues pour `MySQL` 4.0.

`MyODBC 2.50` **(utilise ODBC SDK 2.5) – Gamma**
: En utilisation croissante. Certains problémes sont apparus avec des applications tierces, et indèpendemment du pilote ODBC ou du serveur utilisè.

Tables á restauration automatique `MyISAM` **– Gamma**
: Ce statut ne concerne que le nouveau code du gestionnaire de tables `MyISAM` qui vèrifie si la table a ètè correctement fermèe lors de l'ouverture, et qui exècute automatiquement la vèrification et rèparation èventuelles de la table.

Insertions de masse – Alpha
: Nouvelle fonctionnalitè des tables `MyISAM` pour `MySQL` 4.0 qui permet des insertions plus rapides.

Verrouillage – Gamma
: Cette fonctionnalitè est trés dèpendante du systéme. Sur certains systémes, il y a de gros problémes lors de l'utilisation du verrouillage systéme (avec la fonction `fcntl()`). Dans ces cas, il faut utiliser `mysqld` avec l'option `--skip-external-locking`. Les problémes sont connus sur certaines distributions Linux, et sur SunOS lorsqu'il est utilisè avec des disques en mode NFS.

`MySQL AB` fournit un support de premiére qualitè pour les clients payant, mais les listes de diffusions de `MySQL` sont gènèralement rapides á donner des rèponses aux questions les plus communes. Les bogues sont gènèralement corrigès aussitôt avec un patch. Pour les bogues sèrieux, il y a presque toujours une nouvelle version.

1.2.4 Quelles tailles de tables supporte MySQL ?

`MySQL` version 3.22 a une limite de 4Go par table. Avec le nouveau format de table `MyISAM`, disponible avec `MySQL` version 3.23, la taille maximale des tables a ètè poussèe á 8 millions de teraoctets (2 ^ 63 octets).

Notez, toutefois, que les systémes d'exploitation ont leur propres limites. Voici quelques exemples :

Systéme d'exploitation	**Limite**
Linux-Intel 32 bit	2Go, 4Go ou plus, suivant la version de Linux
Linux-Alpha	8To (?)
Solaris 2.5.1	2Go (4Go possibles avec un patch)
Solaris 2.6	4Go (peut être modifiè avec une option)
Solaris 2.7 Intel	4Go
Solaris 2.7 UltraSPARC	512Go

En Linux 2.2, vous pouvez avoir des tables plus grandes que 2Go en utilisant le patch LFS pour les systémes de fichiers ext2. En Linux 2.4, le patche existe aussi pour ReiserFS.

Cela signifie que les tables `MySQL` sont gènèralement limitèes par le systéme d'exploitation.

Par dèfaut, les tables `MySQL` peuvent atteindre une taille de 4Go. Vous pouvez vèrifier la taille des tables avec la commande `SHOW TABLE STATUS` ou la commande en ligne `myisamchk -dv nom_de_table`. Voir Section 4.5.6 [SHOW], page 267.

Si vous avez besoin de tables plus grandes que 4Go (et que votre systéme d'exploitation le supporte, modifiez les paramétres `AVG_ROW_LENGTH` et `MAX_ROWS` lorsque vous crèez votre

table. Voir Section 6.5.3 [CREATE TABLE], page 504. Vous pouvez aussi les modifier ultèrieurement avec `ALTER TABLE`. Voir Section 6.5.4 [ALTER TABLE], page 512.

Si vos tables sont accessibles uniquement en lecture, vous pouvez aussi utiliser l'utilitaire `myisampack` pour rassembler et compresser plusieurs tables en une seule. `myisampack` compresse gènèralement la table de prés de 50%, ce qui vous augmente d'autant la taille maximale de la table. Voir Section 4.7.4 [`myisampack`], page 297.

Vous pouvez aussi contourner les limites du systéme d'exploitation avec les tables `MyISAM`, en utilisant l'option `RAID`. Voir Section 6.5.3 [CREATE TABLE], page 504.

Une autre solution est d'utiliser la librairie `MERGE`, qui permet de gèrer plusieurs tables comme une seule. Voir Section 7.2 [`MERGE` tables], page 539.

1.2.5 Compatibilitè an 2000

Le `serveur MySQL` lui même n'a aucun probléme de compatibilitè avec l'an 2000 (Y2K) :

- Le `serveur MySQL` utilise les fonctions de date Unix, et n'a aucun probléme avec les dates jusqu'en `2069`; toutes les annèes ècrites en deux chiffres sont supposèes faire partie de l'intervalle allant de `1970` á `2069`, ce qui signifie que si vous stockez la date `01` dans une colonne de type `year`, le `serveur MySQL` la traitera comme `2001`.
- Toutes les fonctions de dates de `MySQL` sont stockèes dans un fichier '`sql/time.cc`', et sont codèes trés soigneusement pour être compatibles avec l'an 2000.
- En `MySQL` version 3.22 et plus rècent, le type de colonne `YEAR` peut stocker les valeurs `0` et de `1901` á `2155` sur un seul octet, tout en affichant 2 ou 4 chiffres.

Vous pouvez rencontrer des problémes avec les applications qui utilisent le `serveur MySQL` sans être compatible avec l'an 2000. Par exemple, les vieilles applications utilisent des valeurs d'annèes sur deux chiffres (ce qui est ambigu), plutôt qu'avec 4 chiffres. Ce probléme peut être complètè par des applications qui utilisent des valeurs telles que `00` ou `99` comme indicateur de donnèes "manquante".

Malheureusement, ces problémes peuvent se rèvèler difficiles á corriger car diffèrentes applications peuvent être ècrites par diffèrents programmeurs, et chacun utilise un jeu diffèrent de conventions et de fonctions de gestion des dates.

Voici une illustration simple qui montre que le `serveur MySQL` n'a aucun probléme avec les dates jusqu'en 2030 :

```
mysql> DROP TABLE IF EXISTS y2k;
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE y2k (date DATE,
    ->                   date_time DATETIME,
    ->                   time_stamp TIMESTAMP);
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO y2k VALUES
    -> ("1998-12-31","1998-12-31 23:59:59",19981231235959),
    -> ("1999-01-01","1999-01-01 00:00:00",19990101000000),
    -> ("1999-09-09","1999-09-09 23:59:59",19990909235959),
```

```
    -> ("2000-01-01","2000-01-01 00:00:00",20000101000000),
    -> ("2000-02-28","2000-02-28 00:00:00",20000228000000),
    -> ("2000-02-29","2000-02-29 00:00:00",20000229000000),
    -> ("2000-03-01","2000-03-01 00:00:00",20000301000000),
    -> ("2000-12-31","2000-12-31 23:59:59",20001231235959),
    -> ("2001-01-01","2001-01-01 00:00:00",20010101000000),
    -> ("2004-12-31","2004-12-31 23:59:59",20041231235959),
    -> ("2005-01-01","2005-01-01 00:00:00",20050101000000),
    -> ("2030-01-01","2030-01-01 00:00:00",20300101000000),
    -> ("2050-01-01","2050-01-01 00:00:00",20500101000000);
Query OK, 13 rows affected (0.01 sec)
Records: 13  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM y2k;
+------------+---------------------+----------------+
| date       | date_time           | time_stamp     |
+------------+---------------------+----------------+
| 1998-12-31 | 1998-12-31 23:59:59 | 19981231235959 |
| 1999-01-01 | 1999-01-01 00:00:00 | 19990101000000 |
| 1999-09-09 | 1999-09-09 23:59:59 | 19990909235959 |
| 2000-01-01 | 2000-01-01 00:00:00 | 20000101000000 |
| 2000-02-28 | 2000-02-28 00:00:00 | 20000228000000 |
| 2000-02-29 | 2000-02-29 00:00:00 | 20000229000000 |
| 2000-03-01 | 2000-03-01 00:00:00 | 20000301000000 |
| 2000-12-31 | 2000-12-31 23:59:59 | 20001231235959 |
| 2001-01-01 | 2001-01-01 00:00:00 | 20010101000000 |
| 2004-12-31 | 2004-12-31 23:59:59 | 20041231235959 |
| 2005-01-01 | 2005-01-01 00:00:00 | 20050101000000 |
| 2030-01-01 | 2030-01-01 00:00:00 | 20300101000000 |
| 2050-01-01 | 2050-01-01 00:00:00 | 00000000000000 |
+------------+---------------------+----------------+
13 rows in set (0.00 sec)
```

Cet exemple montre que les types `DATE` et `DATETIME` ne poseront aucun probléme avec les dates futures (ils gérent les dates jusqu'en 9999).

Le type `TIMESTAMP`, qui est utilisè pour stocker la date courante, est valide jusqu'en `2030-01-01`. `TIMESTAMP` va de `1970` en `2030` sur les machines 32 bits (valeur signèe). Sur les machines 64 bits, il gére les dates jusqu'en `2106` (valeur non signèe).

Même si le `serveur MySQL` est compatible an 2000, il est de votre responsabilitè de fournir des donnèes non ambiguës. Voyez Section 6.2.2.1 [Y2K issues], page 424 pour les régles du `serveur MySQL` pour traiter les dates ambiguës (les donnèes contenant des annèes exprimèes sur deux chiffres).

1.3 Qui est MySQL AB ?

MySQL AB est l'entreprise des fondateurs de MySQL et les principaux dèveloppeurs. A l'origine, MySQL AB a ètè ètablie en Suéde, par David Axmark, Allan Larsson et Michael Monty Widenius.

Tous les dèveloppeurs du serveur MySQL sont employès par l'entreprise. Nous sommes une organisation virtuelle, avec des employès rèpartis dans une douzaine de pays á travers le monde. Nous communiquons intensivement entre nous sur l'Internet tous les jours, et avec nos utilisateurs, fans et partenaires.

Nous nous consacrons au dèveloppement du logiciel MySQL et á la diffusion de notre base de donnèes auprés des nouveaux utilisateurs. MySQL AB est propriètaire des droits du code source de MySQL, du logo MySQL et de la marque de commerce, ainsi que du manuel. Voir Section 1.2 [What-is], page 3.

Les valeurs fondamentales de MySQL tèmoignent de notre implication auprés de MySQL et des Logiciels libres.

Nous souhaitons que la base de donnèes MySQL soit :

- Le meilleur serveur de bases de donnèes dans le monde et le plus rèpandu.
- Le rendre accessible á tous.
- Facile á utiliser.
- En amèlioration constante, tout en restant rapide et sècuritaire.
- Plaisant á utiliser et amèliorer.
- Sans aucun bogue.

MySQL AB et les collaborateurs de MySQL AB :

- Font la promotion des logiciels libres et supportent la communautè des Logiciels libres.
- Sont de bons citoyens.
- Prèférent les partenaires qui partagent nos valeurs et notre ètat d'esprit.
- Rèpondent aux emails et fournissent de l'aide.
- Sont une entreprise virtuelle, qui travaille en rèseau avec les autres.
- Militent contre les brevets logiciels.

Le site web de MySQL (http://www.mysql.com/) fournit les derniéres informations á propos de MySQL et MySQL AB.

1.3.1 Les services et le modéle d'affaire de MySQL AB

Une des questions les plus frèquentes que nous rencontrons est : "*Comment arrivez-vous á vivre en dèveloppant un produit gratuit ?*" Voici comment.

MySQL AB fait des affaires en vendant du support, des services, des licences commerciales et en percevant des royalties. Nous utilisons ces fonds pour le dèveloppement des produits et des affaires de MySQL.

La compagnie est profitable depuis sa conception. En octobre 2001, nous avons acceptè un financement de la part d'un groupe d'investisseurs scandivanes important et de quelques

business angels. Cet investissement est utilisè pour consolider notre modéle d'affaires et assurer les bases de notre croissance á long terme.

1.3.1.1 Support

MySQL AB est dirigè par ses propriètaires, qui sont les fondateurs et les principaux dèveloppeurs de la base de donnèes MySQL. Les dèveloppeurs se consacrent au support des utilisateurs et des autres utilisateurs, afin de rester au courant de leurs besoins et leurs problémes. Tout notre support est donnè par des dèveloppeurs qualifiès. Les questions vraiment èpineuses sont ètudièes par Michael Monty Widenius, auteur principal du code du serveur MySQL. Voir Section 1.4.1 [Support], page 16.

Pour plus d'informations et pour commander un support de diffèrents niveaux, voyez http://www.mysql.com/support/ ou contactez notre èquipe de vente á vente@mysql.com.

1.3.1.2 Formation et certification

MySQL AB organise des formations MySQL á travers le monde entier. Nous offrons des cours inter et intra entreprise, adaptès aux besoins spècifiques de chaque sociètè. La formation MySQL est aussi disponible auprés de nos partenaires, les centres de formation certifiès MySQL.

Nos documents de formation utilisent les mêmes exemples de bases de donnèes que notre documentation et nos applications d'exemple, et ils sont toujours mis á jour pour prendre en compte les derniéres versions de MySQL. Nos formateurs sont èpaulès par notre èquipe de dèveloppement pour garantir la qualitè de la formation et le dèveloppement continu des documents de cours. Cela vous assure aussi qu'il n'y aura pas de questions laissèes ouvertes durant les cours.

Suivre nos formations vous permettra d'atteindre tous vos buts avec votre application MySQL. Vous allez aussi :

- Gagner du temps.
- Amèliorer les performances de vos applications.
- Rèduire ou èliminer les besoins en matèriels supplèmentaires, ce qui rèduira vos coûts.
- Amèliorer la sècuritè.
- Amèliorer la satisfaction de vos clients et collaborateurs.
- Vous prèparer á la certification MySQL.

Si vous êtes intèressè par nos formations, en tant que participant potentiel, ou comme partenaire de formation, visitez la section de formation á http://www.mysql.com/training/ ou contactez-nous á : training@mysql.com.

Le programme de certification MySQL est publiè dans le second semestre 2002. Pour plus de dètails, voyez http://www.mysql.com/certification/.

1.3.1.3 Conseil

MySQL AB et ses partenaires accrèditès offrent des services de conseil aux utilisateurs du serveur MySQL et á ceux qui intégrent le serveur MySQL dans leurs logiciels, á travers le monde.

Nos consultants peuvent vous aider á concevoir et paramètrer vos bases, construire des requêtes efficaces, optimiser votre plate-forme, rèsoudre les problémes de migration, installer la rèplication, bâtir des applications transactionnelles robustes et bien plus encore. Nous aidons aussi les clients á intègrer le serveur MySQL dans leurs produits et applications, pour un dèploiement d'envergure.

Nos consultants travaillent en collaboration ètroite avec notre èquipe de dèveloppement pour assurer la qualitè technique de nos services professionnels. Les missions de conseil peuvent aller de sessions de dèmarrage de deux jours á des projets de plusieurs semaines ou mois. Notre expertise couvre non seulement le serveur MySQL, mais s'ètend aussi aux langages de programmation tels que PHP, Perl et d'autres encore.

Si vous êtes interessè par nos services de conseil ou si vous souhaitez devenir un partenaire conseil, visitez la section conseil de notre site web á http://www.mysql.com/consulting/ ou contactez notre èquipe de conseil á consulting@mysql.com.

1.3.1.4 Licences commerciales

La base de donnèes MySQL est publiée sous la licence GNU General Public License (GPL). Cela signifie que le logiciel MySQL peut être utilisè gratuitement, en acceptant les termes de la licence GPL. Si vous ne voulez pas être liè par les termes de la licence GPL (comme le fait que votre application aussi doit être GPL), vous pouvez acheter une licence du même produit auprés de MySQL AB. Voyez http://www.mysql.com/support/arrangements/price.html. Comme MySQL AB est propriètaire du copyright du code source de MySQL, nous pouvons utiliser une double licence, qui fait que le même produit est disponible sous la licence GPL et sous une licence commerciale. Cela ne change en rien l'implication de MySQL AB dans le mouvement des logiciels libres. Pour plus de dètails sur quand une licence commerciale est nècessaire, voyez Section 1.4.3 [MySQL licenses], page 17.

Nous vendons aussi des licences commerciales aux logiciels Open Source GPL qui ajoutent á la valeur du serveur serveur MySQL. Un bon exemple est le gestionnaire de table transactionnel InnoDB qui offre le support ACID, le verrouillage de ligne, la restauration aprés crash, le multi versionnage, le support des clès ètrangéres, etc. Voir Section 7.5 [InnoDB], page 544.

1.3.1.5 Partenariats

MySQL AB a un programme de partenariat mondial qui couvre la formation, le conseil et le support, les publications, la revente et la distribution des produits MySQL. Les partenaires MySQL AB gagnent en visibilitè grâce au site http://www.mysql.com/ et le droit d'utiliser certaines versions spèciales des marques de commerce MySQL pour identifier leurs produits et promouvoir leur entreprise.

Si vous êtes intèressè á devenir un partenaire MySQL AB, envoyez un email á partenaire@mysql.com.

Le mot MySQL et le logo MySQL avec le dauphin sont des marques commerciales de MySQL AB. Voir Section 1.4.4 [MySQL AB Logos and Trademarks], page 19. Ces marques reprèsentent un investissement capital que les fondateurs de MySQL ont placè depuis plusieurs annèes.

1.3.1.6 Publicitè

Le site web MySQL (http://www.mysql.com/) est trés populaire auprés des dèveloppeurs et des utilisateurs. En Octobre 2001, nous avons eu 10 millions de pages vues. Nos visiteurs reprèsentent un groupe effectuant des dècisions d'achat et des recommandations pour le matèriel et les logiciels. Douze pour cents de nos visiteurs èmettent des dècisions d'achat, et seulement 9 % n'en prennent pas du tout. Plus de 65 % d'entre eux ont effectuè un ou plusieurs achats sur internet ces six derniers mois, et 70 % prèvoient d'en faire un dans le mois á venir.

1.3.2 Contacts

Le site web de MySQL (http://www.mysql.com/) fournit les derniéres informations á propos de MySQL et MySQL AB.

Pour les contacts presse et les questions qui ne sont pas couvertes par les annonces officielles (http://www.mysql.com/news/), envoyez un email á press@mysql.com.

Si vous avez un contrat de support valide avec MySQL AB, vous obtiendrez des rèponses rapides et prècises de notre èquipe technique sur le logiciel MySQL. Pour plus d'informations, voyez Section 1.4.1 [Support], page 16. Sur notre site web, voyez http://www.mysql.com/support/, ou envoyez un email á vente@mysql.com.

Pour des informations sur les formations MySQL, visitez la section formation á http://www.mysql.com/training/. Si vous avez un accés restreint á Internet, contactez l'èquipe de formation de MySQL AB á l'adresse training@mysql.com. Voir Section 1.3.1.2 [Business Services Training], page 13.

Pour des informations sur le programme de certification MySQL, voyez la section http://www.mysql.com/certification/. Voir Section 1.3.1.2 [Business Services Training], page 13.

Si vous êtes interessè par du conseil, visitez la section conseil á http://www.mysql.com/consulting/.
Si vous avez un accés restreint á Internet, contactez l'èquip de conseil de MySQL AB á consulting@mysql.com. Voir Section 1.3.1.3 [Business Services Consulting], page 13.

Les licences commerciales peuvent être commandèes en ligne á http://order.mysql.com/.
Vous trouverez aussi des informations sur les commandes par fax á MySQL AB. Plus d'informations sur les licences sont disponibles á http://www.mysql.com/support/arrangements/price.h
Si vous avez des questions qui concernent les licences ou que vous souhaitez un devis pour un achat de nombreuses licences, remplissez le formulaire de contact disponible sur le site web (http://www.mysql.com/) ou envoyez un email á licence@mysql.com (pour les questions de licence) ou á vente@mysql.com (pour les devis). Voir Section 1.4.3 [MySQL licenses], page 17.

Si vous reprèsentez une entreprise qui est interessèe par un partenariat avec MySQL AB, envoyez un email á partenaire@mysql.com. Voir Section 1.3.1.5 [Business Services Partnering], page 14.

Pour plus d'informations sur la politique de marque de commerce de MySQL, reportez-vous á http://www.mysql.com/company/trademark.html ou envoyez un email á trademark@mysql.com. Voir Section 1.4.4 [MySQL AB Logos and Trademarks], page 19.

Si vous êtes interessè par un des emplois á MySQL AB prèsentès dans notre section job (http://www.mysql.com/company/jobs/), envoyez un email á jobs@mysql.com. N'envoyez pas votre CV en piéce jointe, mais collez-le dans le corps du texte á la fin de votre message.

Pour des discussions gènèrales entre utilisateurs, dirigez vos questions sur la liste de diffusion approprièe. Voir Section 1.6.2 [Questions], page 24.

Les rapports d'erreurs (aussi appelès bogues), ainsi que les questions et suggestions doivent être envoyès á la liste de diffusion mysql@lists.mysql.com. Si vous avez trouvè un trou de sècuritè critique dans le serveur MySQL, envoyez votre courriel á security@mysql.com. Voir Section 1.6.2.3 [Bug reports], page 27.

Si vous avez un rèsultat de test que nous pouvons publier, contactez-nous á benchmarks@mysql.com.

Si vous avez des suggestions concernant les amèliorations ou les corrections de ce manuel, envoyez-les á l'èquipe du manuel docs@mysql.com. Pour les remarques spècifiques á la version française, vous pouvez aussi contacter directement Damien Seguy á damien.seguy@nexen.net.

Pour les questions ou commentaires á propos du fonctionnement ou du contenu du site web MySQL (http://www.mysql.com/), envoyez un email á webmaster@mysql.com.

Pour les questions á propos des portails MySQL (http://www.mysql.com/portal/), envoyez un email á portals@mysql.com.

MySQL AB a une politique de protection des donnèes privèes qui est prèsentèe á http://www.mysql.com/company/privacy.html. Pour toutes les questions concernant cette politique, envoyez un email á privacy@mysql.com.

Pour toutes les autres questions, envoyez un email á info@mysql.com.

1.4 Support MySQL et licences

Cette section dècrit le support MySQL et les accords de licence.

1.4.1 Support proposè par MySQL AB

Le support technique de MySQL AB est la rèponse individualisèe á vos problémes particuliers, en direct, de l'èquipe d'ingènieurs qui programme la base de donnèes MySQL.

Nous tâchons d'avoir un support technique exhaustif et global. Presque tous les problémes qui impliquent MySQL sont importants pour nous, s'ils sont importants pour vous. Typiquement, les clients qui recherchent de l'aide sur les diffèrentes commandes, qui souhaitent rèsoudre des problémes de performance, rèparer des systémes corrompus, comprendre les impacts des systémes d'exploitation ou des rèseaux sur les performances de MySQL, mettre en place des bonnes pratiques pour la sauvegarde et l'entretien, utiliser les APIs, etc. Notre support couvre uniquement le seveur MySQL et nos propres utilitaires, pas les produits tiers d'entreprises qui accédent au serveur MySQL, même si nous pouvons parfois aider.

Les informations dètaillèes sur les diffèrents niveaux de support sont disponibles á http://www.mysql.com/support/, et les contrats de support peuvent aussi être

commandès en ligne. Si vous avez un accés restreint á Internet, vous pouvez contacter notre èquipe commerciale á vente@mysql.com.

Le support technique est identique á l'assurance vie. Vous pouvez vivre trés heureux sans pendant plusieurs annèes, mais lorsque vous rencontrerez une catastrophe, il sera trop tard pour l'acheter. Si vous utilisez MySQL pour des applications importantes et que vous rencontrez des problémes, cela peut vous prendre trés longtemps pour comprendre par vous- même. Vous pourriez alors avoir besoin de communiquer avec les techniciens les plus expèrimentès de MySQL, ceux qui sont employès par MySQL AB.

1.4.2 Copyrights et licences utilisèes par MySQL

MySQL AB est propriètaire du copyright du code source de MySQL, des logos MySQL, de la marque de commerce et de ce manuel. Voir Section 1.3 [What is MySQL AB], page 12. Plusieurs licences distinctes sont disponibles pour la diffusion de MySQL :

1. Toutes les sources spècifiques de MySQL pour le serveur, le client mysqlclient et la librairie, ainsi que la librairie GNU readline sont couverts par la licence GNU General Public License. Voir Annexe H [GPL license], page 832. Le texte de cette licence est aussi disponible dans le fichier intitulè 'COPYING' dans les distributions.
2. La librairie GNU getopt est sous la licence GNU Lesser General Public License. Voir Annexe I [LGPL license], page 841.
3. Certaines parties du code source (la librairie regexp) sont placèes sous un copyright de type Berkeley.
4. Les anciennes versions de MySQL (3.22 et plus rècentes) sont sujettes á des licences plus strictes (http://www.mysql.com/support/arrangements/mypl.html). Voyez la documentation spècifique de chaque version pour plus de dètails.
5. La manuel n'est actuellement **PAS** distribuè sous la licence GPL. L'utilisation du manuel est sujette á ces conditions :
   - Les conversions en d'autres formats sont autorisèes, mais le contenu ne doit pas être modifiè ou èditè de quelque maniére que ce soit.
   - Vous pouvez crèer une version imprimèe pour votre utilisation personnelle.
   - Pour toutes les autres utilisations, telles que la vente de versions imprimèes, ou l'utilisation (en partie ou en totalitè) du manuel dans d'autres publications, un accord doit être conclu au prèalable avec MySQL AB.

   Communiquez par e-mail avec docs@mysql.com pour plus d'informations, ou si vous êtes interessè par la traduction du manuel.

Pour plus de dètails sur comment les licences MySQL fonctionnent, voyez Section 1.4.3 [MySQL licenses], page 17. Voyez aussi Section 1.4.4 [MySQL AB Logos and Trademarks], page 19.

1.4.3 Licences MySQL

Le logiciel MySQL est publiè sous la licence GNU General Public License (GPL), qui est probablement mieux connue sous le nom de Open Source. Les termes exacts de la

licence GPL sont disponibles sur le site de http://www.gnu.org/licenses/. Voyez aussi http://www.gnu.org/licenses/gpl-faq.html et http://www.gnu.org/philosophy/enforcing-gpl.ht

Comme le logiciel MySQL est publiè sous la licence GPL, il est souvent utilisè gratuitement, mais pour certains usages vous souhaiterez peut-être acheter une licence commerciale auprés de MySQL AB á http://order.mysql.com/. Voyez http://www.mysql.com/support/arrangements.html pour plus d'informations.

Les anciennes versions de MySQL (3.22 et plus anciennes) sont sujettes á une licence plus stricte (http://www.mysql.com/support/arrangements/mypl.html). Voyez la documentation spècifique de chaque version pour information.

Notez bien que l'utilisation du logiciel MySQL sous une licence commerciale. GPL, ou toute autre ancienne licence MySQL ne vous donne pas automatiquement le droit d'utiliser les marques commerciales de MySQL AB. Voir Section 1.4.4 [MySQL AB Logos and Trademarks], page 19.

1.4.3.1 Utiliser MySQL avec la licence commerciale

La licence GPL est contagieuse, dans le sens oú lorsqu'un programme est liè á la licence GPL, toutes les sources de toutes les parties du produit final doivent aussi être publièe sous la licence GPL. Sinon, vous violez la licence, et annulez vos droits d'utiliser le programme GPL.

Vous avez besoin d'une licence commerciale pour :

- Lorsque vous reliez un programme á un logiciel GPL issu de MySQL et que vous ne voulez pas que le produit final soit publiè sous la licence GPL, peut être parce que vous souhaitez publier un produit commercial ou conserver du code non-GPL pour d'autres raisons. Lorsque vous achetez une licence commerciale vous n'utilisez plus MySQL sous la licence GPL, même si c'est le même code.
- Lorsque vous distribuez une application non-GPL qui fonctionne **uniquement** avec MySQL, et que vous distribuez cette application avec MySQL. Ce type de solution est considèrè comme un lien, même si c'est fait via le rèseau.
- Lorsque vous distribuez des copies de MySQL sans fournir le code source original, comme requis par la licence GPL.
- Lorsque vous voulez supporter le dèveloppement de MySQL même si vous n'avez pas besoin formellement de la licence commerciale. Acheter du support auprés de MySQL AB est une autre bonne solution pour contribuer au dèveloppement de MySQL, avec des avantages directs pour vous. Voir Section 1.4.1 [Support], page 16.

Si vous avez besoin d'une licence, vous en aurez besoin d'une pour chaque installation de MySQL. Cela est valable quelque soit le nombre de processeurs de la machine, et il n'y a pas de limite artificielle de nombre de connexion simultanèes.

Pour les licences commerciales, voyez notre site web http://www.mysql.com/support/arrangements/pric
Pour les contrats de support, voyez http://www.mysql.com/support/. Si vous avez des besoin spèciaux, ou que vous avez un accés restreint á internet, contactez notre èquipe de vente á vente@mysql.com.

1.4.3.2 Utiliser MySQL sous licence GPL libre

Vous pouvez utiliser le logiciel MySQL sous la licence gratuite GPL si vous acceptez les termes et conditions de la licence GPL. Pour une liste exhaustive des questions courantes á propos de la licence GPL, voyez la FAQ gènèrale de la Free Software Foundation á http://www.gnu.org/licenses/gpl-faq.html. Quelques cas courants :

- Lorsque vous distribuez votre application et le code source MySQL, ensemble avec la licence GPL.
- Lorsque vous distribuez le code source MySQL avec d'autres programmes qui ne sont pas liès ou dèpendant de MySQL pour leur fonctionnalitès, même si vous vendez la distribution commercialement. Ceci s'appelle une agrègation simple en termes GPL.
- Si vous ne distribuez **aucune** partie de MySQL, vous pouvez l'utiliser gratuitement.
- Lorsque vous êtes un fournisseur de services Internet (ISP), qui offre un hèbergement web avec le MySQL. Cependant, nous vous encourageons á faire affaires avec un ISP qui dispose du support MySQL, car cela vous donne l'assurance que si votre hèbergeur a des problémes avec l'installation de MySQL, votre hèbergeur aura les ressources pour rèsoudre votre probléme. Notez que si votre ISP n'a pas de licence commerciale pour le le MySQL, il devrait au moins fournir l'accés en lecture aux sources aux clients pour qu'ils vèrifient que l'installation de MySQL est correctement patchèe.
- Lorsque vous utilisez le serveur de bases de donnèes MySQL en conjonction avec un serveur web, vous n'avez pas besoin de licence commerciale (aussi longtemps que ce n'est pas un produit que vous distribuez). Ceci est même vrai si vous avec un serveur web qui utilise le serveur MySQL, car vous ne distribuez pas de partie de MySQL. Cependant, dans ce cas, nous vous demandons d'acheter du support MySQL car MySQL aide votre entreprise

Si vous utilisez le serveur de bases de donnèes MySQL sans avoir besoin de licence commerciale, nous vous encourageons á acheter le support auprés de MySQL AB malgrè tout. De cette façon, vous contribuez au dèveloppement de MySQL et en obtenez des avantages immèdiats vous-même. Voir Section 1.4.1 [Support], page 16.

Si vous utilisez MySQL dans un contexte commercial tel que vous en tirez profit, nous vous demandons de participer au dèveloppement de MySQL en achetant du support. Nous pensons que comme MySQL aide votre entreprise, il est raisonnable de vous demander d'aider á votre tour MySQL AB. (Sinon, lorsque vous nous posez des questions de support, non seulement vous utilisez gratuitement un systéme dans lequel nous avons investi beaucoup de temps, mais en plus, vous nous demandez du support gratuit, en plus !)

1.4.4 Logos MySQL AB et marque dèposèe

Beaucoup d'utilisateurs de MySQL souhaitent afficher le logo du dauphin MySQL AB sur leur sites web, leur livres ou leurs produits. Nous encourageons ces actes, tant qu'on part du principe que le mot MySQL et le logo du dauphin MySQL sont des marques dèposèes par MySQL AB et ne doivent être utilisès que dans les conditions dècrites á la page suivante : http://www.mysql.com/company/trademark.html.

1.4.4.1 Le logo original de MySQL

Le dauphin MySQL a ètè conçu par l'agence de publicitè finlandaise Priority, en 2001. Le dauphin a ètè choisi en tant que symbole reprèsentatif de MySQL, car c'est un animal intelligent, rapide et èlancè, qui parcourt sans effort l'ocèan des donnèes. Nous aimons aussi les dauphins.

Le logo original de MySQL ne peut être utilisè que par les reprèsentants de MySQL AB et par ceux qui ont un accord signè leur permettant de le faire.

1.4.4.2 Logos MySQL qui peuvent être utilisès dans autorisation prèalable

Nous avons conçu un jeu de logos qui peuvent être utilisès *sous conditions*, et qui peuvent être tèlèchargès depuis notre site web á http://www.mysql.com/press/logos.html et utilisès sur les sites web tiers sans autorisation ècrite de MySQL AB. L'utilisation de ces logos n'est pas inconditionnelle, mais, comme leur nom l'implique, sujette á notre politique de marque de commerce qui est aussi disponible sur notre site web. Il est recommandè de lire ce document avant d'utiliser les logos sur votre site web. En bref, les prè-requis sont :

- Utilisez le logo que vous souhaitez tel que prèsentè sur le site web http://www.mysql.com/. Vous pouvez l'ètendre ou le rèduire suivant vos besoins, mais ne modifier ni les couleurs, ni le design, ni le graphisme en aucune façon.
- Faîtes en sorte qu'il soit èvident que vous, et non pas MySQL AB, êtes le crèateur de ce site, qui arbore les couleurs de MySQL.
- N'utilisez pas le logo d'une façon qui porte prèjudice á MySQL AB ou aux valeurs de MySQL AB. Nous nous rèservons le droit de retirer notre autorisation d'utiliser le logo de MySQL AB.
- Si vous utilisez le logo sur un site web, rendez-le clickable, avec un lien sur le site http://www.mysql.com/.
- Si vous utilisez le serveur web MySQL sous licence GPL dans votre application, votre application doit porter la mention Open Source et être capable de se connecter á un serveur MySQL.

Contactez-nous á trademark@mysql.com pour voir avec nous tous les arrangements spèciaux qui vous conviendraient.

1.4.4.3 Quand avez vous besoin d'autorisation pour utiliser le logo MySQL?

Dans les cas suivants vous devez obtenir une permission ècrite de MySQL AB avant d'utiliser les logos MySQL :

- Lorsque vous affichez un des logos MySQL AB n'importe oú, en dehors de votre site web.
- Lorsque vous affichez l'un des logos MySQL AB hormis les logos á *utilisation conditionnèe* mentionnès auparavant sur le site web, ou ailleurs

En dehors des raisons commerciales et lègales, nous nous devons de suivre l'utilisation du logo MySQL sur les produits, livres, etc. Nous demanderons une compensation pour

l'affichage des logos MySQL AB sur les produits, car nous pensons qu'il est raisonnable qu'une partie des revenus ainsi gènèrès servent á poursuivre le dèveloppement de la base de donnèes MySQL.

1.4.4.4 Logos de partenariat MySQL AB

Le logo partenariat MySQL ne doit être utilisè que par les compagnies et les personnes ayant un partenariat ècrit avec MySQL AB Le partenariat inclut une certification en tant que consultant ou professeur MySQL.

Merci de visiter Section 1.3.1.5 [Partnering], page 14.

1.4.4.5 Utiliser le nom MySQL sur des documents imprimès ou des prèsentations

MySQL AB apprècie les rèfèrences á la base MySQL. Il faut toutefois garder á l'esprit que MySQL est une marque commerciale, propriètè de MySQL AB. A cause de cela, il faut ajouter au logo le symbole de marque de commerce (TM) lors de la premiére ou de la plus visible utilisation du mot MySQL dans le texte et, lá oú appropriè, ètablir clairement que MySQL est une marque commerciale de MySQL AB. Reportez-vous á notre politique de marque commerciale á http://www.mysql.com/company/trademark.html pour plus de dètails.

1.4.4.6 Utilisation du nom MySQL dans un nom de sociètè ou de produit

L'utilisation du mot MySQL dans le nom d'une compagnie, d'un produit ou d'un nom de domaine Internet est interdite sans une permission ècrite de MySQL AB.

1.5 MySQL 4.x In A Nutshell

Longtemps promis par MySQL AB et longtemps attendu par nos utilisateurs, le serveur MySQL 4.0 est maintenant disponible en version bèta pour le tèlèchargement á partir de http://www.mysql.com/ et de nos miroirs.

La plus part des nouvelles fonctionnalitès du serveur MySQL 4.0 sont adaptèes á nos affaires courantes et utilisateurs dèveloppant le logiciel de bases de donnèes MySQL en tant que solution pour les tâches critiques et les bases de donnèes á lourdes charges. Les autres nouveautès ciblent les utilisateurs de bases de donnèes intègrèes.

1.5.1 Phases de publication

La publication du serveur MySQL 4.x se dèroule en plusieurs ètapes, en commencant par la premiére version, nommèe 4.0.0-alpha, qui contient dèjá la plupart des nouvelles fonctionnalitès. Des fonctionnalitès supplèmentaires ont ètè ajoutèes en MySQL 4.0.1, 4.0.2, et ainsi de suite ; MySQL 4.0.3 a ètè dèclarèe beta. D'autres fonctionnalitès seront ajoutèes en MySQL 4.1, destinèe á la publication en phase alpha á la fin de 2002.

1.5.2 Utilisation immèdiate en production

Il est recommandè aux utilisateurs de ne pas passer leurs systémes en production sous le serveur MySQL 4.x, jusqu'á ce qu'elle soit publièe en phase bèta (telle que la 4.0.3 beta). Toutefois, même la version initiale a passè avec succés notre batterie de tests, sans aucune erreur sur aucune plate-forme que nous avons utilisè. Etant donnè le grand nombre de fonctionalitès supplèmentaires, nous recommandons le serveur MySQL, même en version alpha, pour les phases de dèveloppement. L'agenda de publication du serveur MySQL 4.x est tel qu'il atteindra un ètat stable avant les applications qui sont aujourd'hui en phase de dèveloppement.

1.5.3 MySQL intègrè

`libmysqld` rend le serveur MySQL parfaitement utilisable pour une vaste gamme d'applications. En utilisant la librairie intègrèe MySQL, vous pouvez intègrer le serveur MySQL dans diverses applications et appareils èlectroniques, oú l'utilisateur final n'aura aucune visibilitè sur les systémes sous-jacents. Intègrer MySQL est idèal pour les applications de back office dans les systémes Internet, les bornes publiques, les serveurs web hautes performances, les bases de donnèes distribuèes sur CD ROM, etc.

De nombreux utilisateurs de `libmysqld` profitent de la *double licence* de MySQL. Pour ceux qui ne souhaitent pas être liès par la licence GPL, le logiciel est aussi disponible avec une licence commerciale. La librairie intègrèe MySQL utilise la même interface que le client normal, ce qui la rend pratique et facile á utiliser. Voir Section 8.4.9 [libmysqld], page 662.

1.5.4 Autres nouveautès de MySQL 4.0

- La version 4.0 amèliore la *vitesse du serveur MySQL* dans de nombreuses situations, comme les `INSERT` de masse, les recherches sur les index compactès, la crèation d'index en texte plein, ainsi que sur les dènombrements de lignes distinctes.
- Le gestionnaire de table `InnoDB` est proposè comme gestionnaire standard du serveur MySQL, incluant le support complet des `transactions` et le `verrouillage des lignes`.
- Nos utilisateurs allemands, autrichiens et suisses noteront que leur jeu de caractéres, le `latin1_de`, gére correctement les *tri en allemand*, plaçant les umlauts allemands dans le même ordre que celui de l'annuaire.
- La migration depuis les autres systémes de bases de donnèes vers MySQL ont ètè simplifiès pour inclure `TRUNCATE TABLE` (comme sous Oracle) et `IDENTITY` comme synonymes pour les clès automatiquement incrèmentès (comme sous Sybase). De nombreux utilisateurs apprècieront que MySQL supporte dèsormais la commande `UNION`, une fonctionnalitè longtemps attendue.
- En mettant en place de nouvelles fonctionnalitès pour les utilisateurs, nous n'avons pas oubliè la communautè de nos utilisateurs fidéles. Nous avons dèsormais des commandes `DELETE` et `UPDATE` multi tables. En ajoutant le support des `liens symboliques` á `MyISAM` au niveau des tables (et non pas au niveau des bases, comme avant), ainsi qu'en supportant les liens symboliques par dèfaut sous Windows, nous espèrons montrer que

nous prenons á coeur les demandes d'amèliorations. Les fonctions comme `SQL_CALC_FOUND_ROWS` et `FOUND_ROWS()` permettent de savoir combien de lignes une requête sans clause `LIMIT` aurait retournè.

1.5.5 Fonctionnalitès á venir de MySQL 4.x

Vous pouvez anticiper les fonctionnalitès suivantes dans les prochaines versions de MySQL Server 4.x :

- Les utilisateurs exigeants du serveur MySQL apprècieront les nouveautès de notre systéme de rèplication et des sauvegardes en ligne, á chaud. Les versions suivantes incluront aussi la `replication garantie`; dèjá en place dans la version 4.0, la commande `LOAD DATA FROM MASTER` èvoluera rapidement vers une configuration automatique. Les `sauvegardes en ligne` rendront simples les rèplications, sans interrompre le serveur principal, et auront peu d'impact sur les performances des systémes sous forte charge.
- Une fonctionnalitè pratique pour les administrateurs de bases donnèes sera la possibilitè de modifier les paramétres de dèmarrage de `mysqld` sans interrompre le serveur.
- Les nouvelles fonctionnalitès de recherche en texte plein (`FULLTEXT`) du serveur MySQL 4.0 permettent l'indexation de grandes quantitès de texte avec des logiques binaires ou humaines de recherche. Les utilisateurs peuvent paramètrer la taille de mot minimale et dèfinir leur propre liste de mots ignorès. C'est tout une nouvelle gamme d'applications qui seront possibles avec le serveur MySQL.
- De nombreuses applications exploitant la base en lecture bènèficieront d'une augmentation de vitesse grâce á la rèècriture du cache des clès.
- De nombreux dèveloppeurs apprècieront la nouvelle commande `d'aide MySQL` dans le client.

1.5.6 MySQL 4.1 : Les nouvelles fonctionnalitès

Le serveur MySQL 4.0 dèfinit les bases des nouvelles fonctionnalitès des versions 4.1 et plus de MySQL, comme les `requêtes imbriquèes` (4.1), les `procèdures stockèes` (5.0), et les `régles d'intègritè des clefs ètrangéres` pour les tables au format `MyISAM` (5.0), ce qui forme la partie haute de la liste de souhaits de plusieurs de nos utilisateurs.

Aprés ces additions, la critique devra être plus imaginative pour trouver les dèficiences du systéme de gestion de bases de donnèes MySQL. Dèjá connu depuis longtemps pour sa stabilitè, sa rapiditè et sa facilitè de prise en main, MySQL remplira pleinement les attentes de tous les acheteurs exigeants.

1.6 Sources d'informations MySQL

1.6.1 Portails MySQL

Les `portails MySQL` (`http://www.mysql.com/portal/`) reprèsentent l'annuaire le plus complet pour trouver des `partenaires MySQL AB`, ainsi que des livres ou des solutions

MySQL. Les solutions sont classèes et rangèes en ordre d'importance, pour que vous puissiez facilement trouver les informations.

En vous enregistrant en tant qu'utilisateur, vous gagnez la possibilitè de commenter et noter les solutions du portail. Vous recevrez aussi des lettres d'actualitè suivant votre profil d'utilisateur, qui est modifiable á tout moment.

Les catègories actuelles du portail MySQL incluent :

Partenaires
Trouver des partenaires MySQL AB dans le monde.

Livres Commenter, voter et acheter des livres concernant MySQL.

Dèveloppement
Diffèrents liens vers des sites web qui utilisent le serveur MySQL dans diffèrentes situations, avec une description pour chaque site. Ces informations vous donneront une idèe de qui utilise MySQL et comment le serveur MySQL peut vous satisfaire.

Parlez-nous de *votre* site ou de vos rèussites ! Visitez http://www.mysql.com/feedback/testi

Logiciels Trouver, acheter et tèlècharger diffèrentes applications et complèments á utiliser avec le serveur MySQL.

Distributions
Depuis ce site, vous pouvez trouver les diffèrentes distributions Linux et d'autres logiciels qui contiennent MySQL.

Fournisseurs de services
Les entreprises qui proposent des services autour de MySQL.

1.6.2 Listes de diffusion MySQL

Cette section vous prèsente les listes de diffusions MySQL, et donne des conseils quand á leur utilisation. En vous inscrivant á une des listes de diffusion, vous recevrez les messages que les autres auront envoyè, et vous pourrez envoyer vos propres questions et rèponses.

1.6.2.1 Les listes de diffusions de MySQL

Pour vous inscrire á la liste de diffusion principale de MySQL, envoyez un courrier èlectronique á mysql-subscribe@lists.mysql.com.

Pour vous dèsinscrire á la liste de diffusion principale de MySQL, envoyez un courrier èlectronique á mysql-unsubscribe@lists.mysql.com.

Seule l'adresse d'oú vous envoyez votre message est importante. La ligne de sujet et le corps sont ignorès.

Si votre adresse de rèponse n'est pas valide, vous pouvez spècifier votre adresse explicitement, en ajoutant un tiret aprés le mot subscribe ou unsubscribe, suivi de votre adresse dans laquelle vous aurez remplacè le caractére '@' par '='. Par exemple, pour inscrire l'adresse your_name@host.domain, envoyez un message á mysql-subscribe-your_name=host.domain@lists.mysql.com.

Les mails envoyès á `mysql-subscribe@lists.mysql.com` et `mysql-unsubscribe@lists.mysql.com` sont gèrè automatiquement par le robot ezmlm. Des informations sur ezmlm sont disponibles á `http://www.ezmlm.org/`.

Pour poster un message sur la liste elle-même, envoyez votre message á `mysql@lists.mysql.com`. Toutefois, **n'envoyez pas** de mail d'inscription ou de dèsinscription á `mysql@lists.mysql.com` car tous les mails envoyès á cette adresse sont distribuès automatiquement á des milliers d'utilisateurs.

Votre site local peut avoir beaucoup d'inscrits á la liste `mysql@lists.mysql.com`. Si c'est le cas, vous pouvez avoir une liste de diffusion locale, de façon á ce que les messages envoyès par `lists.mysql.com` á votre site local soit propagès par votre serveur local. Dans ce cas, contactez votre administrateur local pour être ajoutè ou retirè de la liste.

Si vous voulez que le trafic de cette liste soit envoyè á une autre boîte aux lettres de votre client mail, installez un filtre basè sur les entêtes du message. Vous pouvez utiliser notamment les entêtes `List-ID:` et `Delivered-To:` pour identifier les messages de la liste.

Les listes de diffusion MySQL suivantes existent :

`announce-subscribe@lists.mysql.com` `announce`
: Ceci est la liste de diffusion d'annonces des versions de MySQL et des programmes compagnons. C'est une liste á faible volume, et tout utilisateur doit y être inscrit.

`mysql-subscribe@lists.mysql.com` `mysql`
: La liste de diffusion principale pour les discussions gènèrales sur MySQL. Notez que certains sujets sont á diriger sur les listes spècialisèes. Si vous postez sur la mauvaise liste, vous pourriez ne pas avoir avoir de rèponse.

`mysql-digest-subscribe@lists.mysql.com` `mysql-digest`
: La liste `mysql` en format journalier. Cela signifie que vous recevrez tous les messages de la journèe en un seul gros email.

`bugs-subscribe@lists.mysql.com` `bugs`
: Sur cette liste, vous ne devriez envoyez que des bogues complets, reproductibles ainsi que le rapport qui va avec, en utilisant le script `mysqlbug` (si vous utilisez Windows, il faut aussi inclure la description du systéme d'exploitation et la version de MySQL MySQL). De prèfèrence, vous devriez tester le probléme avec la derniére version stable ou de dèveloppement du serveur MySQL avant de l'envoyer. Tout le monde doit être capable de reproduire le bogue simplement avec la ligne de commande `mysql test < script` avec le cas de test inclus. Tous les bogues doivent être postè sur la liste, et seront corrigès ou documentès dans la prochaine version de MySQL!. Si les modifications sont trop petites, nous publierons aussi un patch qui rèsout le probléme.

`bugs-digest-subscribe@lists.mysql.com` `bugs-digest`
: La liste `bugs` en format journalier.

`internals-subscribe@lists.mysql.com` `internals`
: Une liste pour ceux qui travaillent sur le code MySQL. Sur cette liste, vous pouvez discuter du dèveloppement de MySQL et envoyer des patchs.

`internals-digest-subscribe@lists.mysql.com` `internals-digest`
: La liste `internals` en format journalier.

`java-subscribe@lists.mysql.com` `java`
: Une liste pour ceux qui utilisent MySQL et java. Elle concerne majoritairement les pilotes JDBC.

`java-digest-subscribe@lists.mysql.com` `java-digest`
: La liste `java` en format journalier.

`win32-subscribe@lists.mysql.com` `win32`
: Une liste pour ceux qui utilisent MySQL sur les systémes d'exploitation de Microsoft, tels que Windows 9x/Me/NT/2000/XP.

`win32-digest-subscribe@lists.mysql.com` `win32-digest`
: La liste `win32` en format journalier.

`myodbc-subscribe@lists.mysql.com` `myodbc`
: Une liste pour tout ce qui concerne la connexion á MySQL avec le pilote ODBC.

`myodbc-digest-subscribe@lists.mysql.com` `myodbc-digest`
: La liste `myodbc` en format journalier.

`mysqlcc-subscribe@lists.mysql.com` `mysqlcc`
: Une liste pour tout ce qui concerne le client graphique `MySQL Control Center`.

`mysqlcc-digest-subscribe@lists.mysql.com` `mysqlcc-digest`
: La liste `mysqlcc` en format journalier.

`plusplus-subscribe@lists.mysql.com` `plusplus`
: Une liste pour tout ce qui concerne la programmation avec les API C++ de MySQL.

`plusplus-digest-subscribe@lists.mysql.com` `plusplus-digest`
: La liste `plusplus` en format journalier.

`msql-mysql-modules-subscribe@lists.mysql.com` `msql-mysql-modules`
: Une liste pour tout ce qui concerne Perl et le support du module msql-mysql-modules.

`msql-mysql-modules-digest-subscribe@lists.mysql.com`
`msql-mysql-modules-digest`
: La liste `msql-mysql-modules` en format journalier.

Vous pouvez vous inscrire ou vous dèsinscrire de toutes les listes en même temps de la même façon que nous l'avons dècrit au dèbut. Dans votre message d'inscription, utilisez simplement le nom de liste appropriè. Par exemple, pour vous inscrire á la liste `myodbc`, envoyez un message á `myodbc-subscribe@lists.mysql.com` ou `myodbc-unsubscribe@lists.mysql.com`.

Si vous ne pouvez pas obtenir d'informations sur la liste de diffusion, une de vos options est de prendre un contrat de support auprés de MySQL AB, qui vous donnera un contact direct avec les dèveloppeurs MySQL. Voir Section 1.4.1 [Support], page 16.

Le tableau suivant prèsente diverses autres listes de diffusions consacrèe á MySQL, dans d'autres langues que l'anglais. Notez que ces ressources ne sont pas gèrèes par MySQL AB, ce qui fait que nous ne pouvons pas garantir leur qualitè.

`mysql-france-subscribe@yahoogroups.com` Une liste de diffusion française
`list@tinc.net` Une liste de diffusion corèenne
: Envoyez un message á`subscribe mysql your@e-mail.address`.

`mysql-de-request@lists.4t2.com` Une liste de diffusion allemande
: Envoyez un message á `subscribe mysql-de your@e-mail.address`. Vous aurez plus d'informations sur cette liste á `http://www.4t2.com/mysql/`.

`mysql-br-request@listas.linkway.com.br` Une liste de diffusion portugaise
: Envoyez un message á `subscribe mysql-br your@e-mail.address`.

`mysql-alta@elistas.net` Une liste de diffusion espagnole
: Envoyez un message á `subscribe mysql your@e-mail.address`.

1.6.2.2 Poser des questions ou rapporter un bogue

Avant de soumettre un rapport de bogue ou une question, commencez par ces ètapes simples :

- Etudiez le manuel MySQL et faites y une recherche á :
  `http://www.mysql.com/doc/`
  Nous nous efforçons de mettre á jour le manuel frèquemment, en y ajoutant les solutions aux nouveaux problémes. L'historique de modification (`http://www.mysql.com/doc/en/News.html`) est particuliérement pratique car il est possible qu'une nouvelle version de MySQL propose dèjá la solution á votre probléme.
- Recherchez dans les archives des listes de diffusion de MySQL :
  `http://lists.mysql.com/`
- Vous pouvez aussi utiliser l'URL `http://www.mysql.com/search/` pour rechercher dans toutes les pages web (y compris le manuel) qui sont situès á `http://www.mysql.com/`.

Si vous n'arrivez pas á trouver une rèponse á votre question dans le manuel ou dans les archives, vèrifiez auprés de votre expert MySQL local. Si vous ne trouvez toujours pas la rèponse, vous pouvez lire la section suivante et envoyer un mail bien prèparè á `mysql@lists.mysql.com`.

1.6.2.3 Comment rapporter un bogue ou un probléme

Ecrire un bon rapport de bogue requiert de la patience, et le faire dés le dèbut èpargnera votre temps et le notre. Un bon rapport de bogue qui contient un cas de test complet amèliorera vos chances de voir le bogue corrigè á la prochaine version. Cette section vous aidera á ècrire correctement un rapport de bogue, de maniére á ce que vous ne gaspillez pas votre temps á faire des textes qui ne nous aideront que peu ou pas.

Nous vous recommandons d'utiliser le script `mysqlbug` pour gènèrer un rapport de bogue (ou rapporter un probléme), dans la mesure du possible. `mysqlbug` est situè dans le dossier '`scripts`' de la distribution, ou, pour les distributions binaire, dans le dossier '`bin`' du

dossier d'installation de MySQL. Si vous êtes dans l'incapacitè d'utiliser `mysqlbug`, vous devez tout de même inclure toutes les informations nècessaires listèes dans cette section.

Le script `mysqlbug` vous aide á gènèrer un rapport en dèterminant automatiquement les informations suivantes, mais si quelque chose d'important lui èchappe, ajoutez le dans votre message! Lisez cette section avec attention, et assurez vous que toutes les informations dècrites ici sont prèsentes dans votre message.

Pour rapporter un bogue ou un probléme, il faut le soumettre á la liste `mysql@lists.mysql.com`. Si vous pouvez rèdiger un cas de test qui dèmontre clairement le bogue, il faut le poster sur la liste `bugs@lists.mysql.com`. Notez que sur cette liste, vous ne devez poster qu'un rapport complet de bogue regènèrable, avec le script `mysqlbug`. Si vous travaillez sous Windows, vous devez ajouter une description de votre systéme d'exploitation et de votre version de MySQL. De prèfèrence, il vaut mieux utiliser la derniére version de MySQL, version stable ou de dèveloppement, avant d'envoyer un message. Tout le monde doit être capable de reproduire le bogue en utilisant simplement la ligne de commande "`mysql test < script`", avec le cas de bogue fourni, ou bien d'exècuter un script shell ou Perl qui est inclut dans le rapport de bogue. Tous les bogues postès sur la liste `bogues` seront corrigès ou documentès dans la prochaine version de MySQL. Si seul de petits changements sont nècessaires, nous publierons aussi un patch.

Si vous avez dècouvert un probléme de sècuritè sensible dans MySQL, il faut envoyer un email á `security@mysql.com`.

Sachez qu'il est toujours possible de rèpondre á un message qui contient trop d'informations, alors qu'il est impossible de rèpondre á um message qui contient trop peu d'informations. Souvent, il est facile d'omettre des faits parce que vous pensez connaître la cause du probléme et supposez que ces dètails ne sont pas importants. Un bon principe á suivre est : si vous avez un doute á propos de quelque chose, faites nous en part. Il est bien plus rapide et bien moins frustrant d'ècrire quelques lignes de plus dans un rapport plutôt que d'être obligè de demander une nouvelle fois et d'attendre une rèponse parce que vous avez oubliè une partie des informations la premiére fois.

L'erreur la plus commune est de ne pas indiquer le numèro de la version de MySQL qui est utilisè, ou de ne pas indiquer le systéme d'exploitation que vous utilisez (y compris le numèro de version de ce systéme d'exploitation). Ce sont des informations de premiére importance, et dans 99% d4es cas, le rapport de bogue est inutilisable sans ces informations. Souvent, nous recevons des questions telles que "Pourquoi est ce que cela ne fonctionne pas pour moi?". Puis nous nous aperçevons que la fonctionnalitès en question n'est même pas programmèe dans la version de MySQL utilisèe, ou que le bogue dècrit est dèjá corrigè dans une nouvelle version de MySQL. Parfois aussi, les erreurs sont dèpendantes des plate-formes. Dans ce cas, il est presque impossible de les corriger sans savoir quel systéme d'exploitation et quelle version exacte est utilisèe.

Pensez aussi á fournir des informations concernant votre compilateur, si c'est pertinent. Souvent, les dèveloppeurs trouvent des bogues dans les compilateurs, et pensent que c'est liès á MySQL. La plupart des compilateurs sont en constant dèveloppement, et s'amèliorent de version en version. Pour dèterminer si votre problée dèpend de votre compilateur, nous avons besoin de savoir quel compilateur est utilisè. Notez que les problémes de compilations sont des bogues, et doivent être traitès avec un rapport de bogues.

Il est particuliérement utile de fournir une bonne description du bogue dans le rapport de bogue. Cela peut être un exemple de ce que vous avez fait qui a conduit au probléme, ou une description prècise. Les meilleurs rapports sont ceux qui incluent un exemple complet permettant de reproduire le bogue. Voir Section E.1.6 [Reproduceable test case], page 820.

Si un programme produit un message d'erreur, il est trés important d'inclure ce message dans votre rapport. Il est prèfèrable que le message soit le message exact, car il est alors possible de le retrouver en utilisant les archives : même la casse doit être respectèe. N'essayez jamais de vous rappeler d'un message d'erreur, mais faites plutôt un copier/coller du message complet dans votre rapport.

Si vous avez un probléme avec MyODBC, essayez de gènèrer un fichier de trace MyODBC. Voir Section 8.3.7 [MyODBC bug report], page 608.

Pensez aussi que de nombreux personnes qui liront votre rapport utilisent un formatage de 80 colonnes. Lorsque vous gènèrez votre rapport et vos exemples avec l'outil de ligne de commande, utilisez une largeur de 80 colonnes. Utilisez l'option `--vertical` (ou la fin de commande `\G`) pour les affichages qui excédent une telle largeur (par exemple, avec la commande `EXPLAIN SELECT`; voyez l'exemple un peu plus tard dans cette section.

Voici un pense-bête des informations á fournir dans votre rapport :

- Le numèro de version de la distribution de MySQL que vous utilisez (par exemple MySQL Version 3.22.22). Vous pouvez connaitre cette version en excutant la commande `mysqladmin version`. `mysqladmin` est situè dans le dossier '`bin`' de votre distribution MySQL.
- Le fabricant et le modéle de votre serveur.
- Le systéme d'exploitation et la version que vous utilisez. Pour la plupart des systémes d'exploitation, vous pouvez obtenir cette information en utilisant la commande Unix command `uname -a`.
- Parfois, la quantitè de mèmoire (physique et virtuelle) est important. Si vous hèsitez, ajoutez la.
- Si vous utilisez une version de MySQL sous forme de source, le nom et le numèro de version du compilateur sont nècessaires. Si vous utilisez une version exècutable, le nom de la distribution est important.
- Si le probléme intervient lors de la compilation, incluez le message d'erreur exact et les quelques lignes de contexte autour du code en question dans le fichier oú il est situè.
- Si `mysqld` s'est arrètè, il est recommandè d'inclure la requête qui a menè á cet arrêt de `mysqld`. Vous pouvez gènèralement la trouver en exècutant `mysqld` en ayant activè les logs. Voir Section E.1.5 [Using log files], page 819.
- Si une table ou une base sont liès au probléme ajoutez le rèsultat de la commande `mysqldump --no-data db_name tbl_name1 tbl_name2 ...`. C'est trés simple á faire, et c'est un moyen efficace d'obtenir un descriptif de table, qui nous permettra de recrèèer une situation comparable á la votre.
- Pour les problémes liès á la vitesse, ou des problémes liès á la commande `SELECT`, pensez á inclure le rèsultat de la commande `EXPLAIN SELECT ...`, et au moins le nombre de ligne que la commande `SELECT` doit produire. Vous devriez aussi inclure le rèsultat de la commande `SHOW CREATE TABLE table_name` pour chaque table impliquèe. Plus vous nous fournirez d'informations, plus nous aurons de chance de vous aider efficacement.

Par exemple, voici un excellent rapport de bogue (postè avec le script `mysqlbug`, et effectivement rèdigè en ANGLAIS) :

Exemple rèalisè avec `mysql` en ligne de commande (notez l'utilisation de la fin de commande `\G`, pour les rèsultats qui pourraient dèpasser les 80 colonnes de large) :

```
mysql> SHOW VARIABLES;
mysql> SHOW COLUMNS FROM ...\G
       <output from SHOW COLUMNS>
mysql> EXPLAIN SELECT ...\G
       <output from EXPLAIN>
mysql> FLUSH STATUS;
mysql> SELECT ...;
       <A short version of the output from SELECT,
       including the time taken to run the query>
mysql> SHOW STATUS;
       <output from SHOW STATUS>
```

- Si un bogue ou un probléme survient lors de l'exècution de `mysqld`, essayez de fournir un script qui reproduit l'anomalie. Ce script doit inclure tous les fichiers sources nècessaires. Plus votre script reproduira fidélement votre situation, le mieux ce sera. Si vous pouvez rèaliser un cas de test postez le sur la liste `bugs@lists.mysql.com` pour un traitement en prioritè!

  Si vous ne pouvez pas fournir de script, fournissez tout au moins le rèsultat de la commande `mysqladmin variables extended-status processlist` dans votre mail pour fournir des informations sur les performances de votre systéme.

- Si vous ne pouvez pas reproduire votre situation en quelques lignes, ou si une table de test est trop grosse á être envoyèe par mail (plus de 10 lignes), exportez vos tables sous forme de fichier avec la commande `mysqldump` et crèez un fichier '`README`' qui dècrit votre probléme.

  Crèez une archive compressèe de votre fichier en utilisant `tar` et `gzip` ou `zip`, et placez le via `ftp` sur le site de `ftp://support.mysql.com/pub/mysql/secret/`. Puis, envoyez une courte description de votre probléme á `bugs@lists.mysql.com`.

- Si vous pensez que le serveur MySQL fournit des rèsultats ètranges pour une requête, incluez non seulement le rèsultat, mais aussi votre propre explication sur ce que le rèsultat devrait être, et un diagnostic de la situation.

- Lorsque vous donnez un exemple du probléme, il est mieux d'utiliser des noms de variables et de tables qui existent dans votre situation, plutôt que d'inventer de nouveaux noms. Le probléme peut être liè au noms des variables ou tables que vous utilisez! Ces cas sont rares, mais il vaut mieux èviter les ambiguitès. Aprés tout, il est plus facile pour vous de fournir un exemple qui utilise votre situation rèelle, et c'est bien mieux pour nous aussi. Si vous avez des donnèes que vous ne souhaitez pas divulguer, vous pouvez utiliser le site `ftp` pour les transfèrer dans le dossier secret `ftp://support.mysql.com/pub/mysql/secret/`. Si les donnèes sont vraiment ultra secrétes et que vous ne souhaitez même pas nous les montrer, alors utilisez d'autres noms et donnèes pour votre rapport, mais considèrez cela comme un dernier recours.

- Incluez toutes les options utilisèes, si possible. Par exemple, indiquez les options que vous utilisez lors du dèmarrage de `mysqld`, et celle que vous utilisez avec les programmes

comme `mysqld` et `mysql`, et le script `configure`, qui sont souvent primordiaux et pertinents. Ce n'est jamais une mauvaise idèe que de les inclure. Si vous utilisez des modules, comme Perl ou PHP, incluez aussi les versions de ces logiciels.

- Si votre question porte sur le systéme de droits, incluez le rèsultat de l'utilitaire `mysqlaccess`, celui de `mysqladmin reload` et tous les messages d'erreurs que vous obtenez lors de la connexion. Lorsque vous testez votre systéme de droits, il faut commencer par utiliser la commande `mysqladmin reload version` et de vous connecter avec le proramme qui vous pose probléme. `mysqlaccess` est situè dans le dossier '`bin`' de votre installation de MySQL.
- Si vous avez un patch pour un bogue, c'est une excellente chose. Mais ne supposez pas que nous n'avons besoin que du patch, ou même que nous allons l'utiliser, si vous ne fournissez pas les informations nècessaires pour le tester. Nous pourrions trouver des problémes gènèrès par votre patch, ou bien nous pourrions ne pas le comprendre du tout. Si tel est le cas, nous ne l'utiliserons pas.

  Si nous ne pouvons pas vèrifier exactement ce pourquoi est fait le patch, nous ne l'utiliserons pas. Les cas de tests seront utiles ici. Montrez nous que votre patch va gènèrer toutes les situations qui pourraient arriver. Si nous trouvons un cas limite dans lequel votre patche ne fonctionne pas, même si il est rare, il risque d'être inutile.
- Les diagnostics sur la nature du bogue, la raison de son dèclenchement ou les effets de bords sont gènèralement faux. Même l'èquipe MySQL ne peut diagnostiquer sans commencer par utiliser un dèbogueur pour dèterminer la cause vèritable.
- Indiquez dans votre mail que vous avez vèrifiè le manuel de rèfèrence et les archives de courrier, de façon á avoir èpuiser les solutions que d'autres avant vous auraient pu trouver.
- Si vous obtenez un message `parse error`, vèrifiez votre syntaxe avec attention. Si vous ne pouvez rien y trouvez á redire, il est trés probable que votre version de MySQL ne supporte pas encore cette fonctionnalitè que vous essayez d'utiliser. Si vous utilisez la version courante de mySQL et que le manuel `http://www.mysql.com/doc/` ne couvre pas la syntaxe que vous utilisez, c'est que MySQL ne supporte pas votre syntaxe. Dans ce cas, vos seules options sont d'implèmenter vous même la syntaxe ou d'envoyez un message á `licence@mysql.com` pour proposer de l'implèmenter.

  Si le manuel prèsente la syntaxe que vous utilisez, mais que vous avez une ancienne version du serveur MySQL, il est recommandè de vèrifier l'historique d'èvolution de MySQL pour savoir quand la syntaxe a ètè supportèe. Dans ce cas, vous avez l'option de mettre á jour votre MySQL avec une version plus rècente. Voir Annexe D [News], page 727.
- Si vous avez un probléme tel que vos donnèes semblent corrompues, ou que vous recevez constamment des errors lors d'accés á une table, vous devriez commener par essayer de rèparer votre table avec l'utilitaire de ligne de commande `myisamchk` ou les syntaxes SQL `CHECK TABLE` et `REPAIR TABLE`. Voir Chapitre 4 [MySQL Database Administration], page 192.
- Si vous avez des tables qui se corrompent facilement, il vous faut essayer de trouver quand et pourquoi cela arrive. Dans ce cas, le fichier '`mysql-data-directory/'hostname'.err`' peut contenir des informations pertinentes qu'il est bon d'inclure dans votre rapport de bogues. Normalement, `mysqld` **ne doit**

jamais corrompre une table si il a ètè interrompu au milieur d'une mise á jour. Si vous pouvez trouvez la cause de l'arrêt de `mysqld`, il est bien plus facile pour nous de fournir un correctif. Voir Section A.1 [What is crashing], page 684.

- Si possible, tèlèchargez et installez la version la plus rècente du serveur MySQL, et vèrifiez si cela rèsoud votre probléme. Toutes les versions de MySQL sont testèes á fond, et doivent fonctionner sans probléme. Nous croyons á la compatibilitè ascendante, et vous devriez pouvoir passer d'une version á l'autre facilement. Voir Section 2.2.4 [Which version], page 76.

Si vous disposez de l'accés au support client, contactez aussi le support client á `mysql-support@mysql.com`, en plus de la liste de rapport de bogues, pour un traitement prioritaire.

Pour des informations sur les rapports de bogues avec `MyODBC`, voyez Section 8.3.4 [ODBC Problems], page 602.

Pour des solutions aux problémes les plus courants, voyez Annexe A [Problems], page 684.

Lorsque des solutions vous sont envoyèes individuellement et non pas á la liste, il est considèrè comme bien vu de rassembler ces rèponses et d'en envoyer un rèsumè sur le liste, de maniére á ce que les autres en profitent aussi.

1.6.2.4 Conseils pour rèpondre sur la liste de diffusion

Si vous pensez que votre rèponse peut avoir un intèrêt gènèral, vous pouvez envisager de l'envoyer sur la liste de diffusion, plutôt que de faire une rèponse personnelle aux demandeurs. Essayez de rendre votre rèponse aussi gènèrale que possible, pour que suffisamment d'autres personnes puissent en profiter. Lorsque vous envoyez une rèponse sur la liste, assurez vous qu'elle ne reprèsente pas un doublon d'une rèponse prècèdente.

Essayez de rèsumer l'essentiel de la question dans votre rèponse. Ne vous croyez pas obligè de citer tout le message original.

Attention : n'envoyez pas de message avec le mode HTML activè ! De nombreux utilisateurs ne lisent pas leurs emails avec un navigateur.

1.7 Quels standards respecte MySQL?

Cette section prèsente comment MySQL interpréte les standards SQL ANSI. Le serveur MySQL dispose de nombreuses extensions au standard ANSI et vous trouverez ici comment les exploiter. Vous trouvrez aussi des informations sur les fonctionnalitès manquantes de MySQL et comment y trouver des palliatifs.

Notre but n'est pas, sans une bonne raison, de restreindre les capacitès de MySQL á un usage unique. Même si nous n'avons pas les ressources de dèveloppement á consacrer á toutes les opportunitès, nous sommes toujours interessès et prêts á aider ceux qui utilisent MySQL dans de nouveaux domaines.

Un de nos objectifs avec ce produit est de tendre á la compatibilitè ANSI 99, mais sans sacrifier la vitesse ou la robustesse. Nous ne reculons pas devant l'ajout de nouvelle fonctionnalitès au langage SQL, ou le support de fonctionnalitès hors SQL, qui amèliorent le

confort d'utilisation de MySQL. La nouvelle interface de gestionnaires `HANDLER` de MySQL 4.0 est un exemple de cette stratègie. Voir Section 6.4.2 [`HANDLER`], page 488.)

Nous continuons de supporter les bases transactionnelles et non transactionnelles pour combler les besoins des sites web ou des applications á fort besoin d'archivage, ainsi que les applications critiques á trés haute disponibilitè.

Le serveur MySQL a ètè conçu pour travailler avec des bases de taille moyenne (de 10 á 100 millions de lignes, ou des tables de 100 Mo) sur des systémes de petite taille. Nous continuons d'amèliorer MySQL pour qu'il fonctionne avec des bases gigantesques (tera octets), tout en conservant la possibilitè de compiler une version rèduite de MySQL pour qu'il fonctionne sur des appareils embarquès ou nomades. L'architecture compacte de MySQL rend possible le support de ces applications si diffèrentes, sans aucun conflit dans les sources.

Nous n'ètudions pas le support du temps rèel ou des bases de donnèes en grappe (même si vous pouvez dores et dèjá rèaliser de nombreuses applications avec les services de rèplication).

Nous ne croyons pas au support natif du XML en base, mais nous allons faire en sorte d'ajouter le support XML que rèclame nos clients du cotè client. Nous pensons qu'il est prèfèrable de conserver le serveur central aussi "simple et efficace" que possible, et dèvelopper les librairies qui gérent la complexitè du cotè client. Cela fait partie de la stratègie que nous avons mentionnè plus tôt, pour ne sacrifier ni la vitesse, ni la robustesse du serveur.

1.7.1 Quels standards suit MySQL ?

SQL92, niveau de base. ODBC niveau 0-3.51.

Nous nous dirigeons vers le support complet du standard ANSI SQL99, mais sans aucune concession sur la vitesse ou la qualitè du code.

1.7.2 Exècuter MySQL en mode ANSI

Si vous dèmarrez `mysqld` avec l'option `--ansi`, les comportements suivants du serveurs MySQL changent :

- `||` devient l'opèrateur de concatènation de chaîne, et non pas l'opèrateur binaire `OR`.
- Vous pouvez ajouter des espaces entre le nom d'une fonction et la parenthése ouvrante sur les arguments : '`(`'. Cela impose le traitement des noms de fonctions comme des mots rèservès.
- '`"`' devient le caractére de protection des identifiants (comme le caractére '`` ` ``' de MySQL) et n'est plus un caractére de limite de chaîne.
- `REAL` est synonyme de `FLOAT` au lieu d'être synonyme de `DOUBLE`.
- Le niveau d'isolation par dèfaut des transactions est `SERIALIZABLE`. Voir Section 6.7.3 [SET TRANSACTION], page 521.

Ceci revient á utiliser les options suivantes : `--sql-mode=REAL_AS_FLOAT,PIPES_AS_CONCAT,ANSI_QUOTES, IGNORE_SPACE,SERIALIZE,ONLY_FULL_GROUP_BY`.

1.7.3 Extensions de MySQL á la norme ANSI SQL92

Le serveur MySQL inclut des extensions que vous ne trouverez probablement pas dans les autres bases de donnèes. Soyez prèvenus que si vous les utilisez, votre code ne sera probablement pas portable sur d'autres serveurs SQL. Dans certains cas, vous pouvez ècrire du code qui inclut des spècificitès de MySQL, mais qui restent portables, en les incluant dans des commentaires de la forme `/*! ... */`. Dans ce cas, le serveur MySQL va analyser la chaîne et exècuter le code á l'intèrieur de ces commentaires comme une commande normale, mais d'autres serveurs ignoreront ces commentaires. Par exemple :

```
SELECT /*! STRAIGHT_JOIN */ col_name FROM table1,table2 WHERE ...
```

Si vous ajoutez le numèro de version aprés le point d'exclamation `'!'`, la syntaxe sera exècutèe uniquement si la version du serveur MySQL est ègale ou plus rècente que le numèro de version utilisè.

```
CREATE /*!32302 TEMPORARY */ TABLE t (a int);
```

Cela signifie que si vous avez la version 3.23.02 ou plus rècente, le serveur MySQL va utiliser le mot rèservè `TEMPORARY`.

Voici une liste des apports spècifiques de MySQL :

- Les types de colonnes `MEDIUMINT`, `SET`, `ENUM` et les types `BLOB` et `TEXT`.
- Les attributs de champs `AUTO_INCREMENT`, `BINARY`, `NULL`, `UNSIGNED` et `ZEROFILL`.
- Toutes les comparaisons de chaînes sont insensibles á la casse par dèfaut, et l'ordre de tri est dèterminè par le jeu de caractéres courant (ISO-8859-1 Latin1 par dèfaut). Si vous en souhaitez un autre, il faut dèclarer les colonnes avec l'attribut `BINARY` ou utiliser l'opèrateur `BINARY` pour forcer les comparaisons á prendre en compte la casse, en fonction du jeu de caractéres utilisè sur l'hôte du serveur MySQL.
- le serveur MySQL fait correspondre á chaque base un dossier dans le systéme de fichiers, et á chaque table, il fait correspondre un fichier, placè dans le dossier de base.

  Ceci a quelques consèquences :

  - Les noms des bases de donnèes et des tables sont sensibles á la casse sur les systémes d'exploitation qui ont des systémes de fichiers sensibles á la casse (comme la plupart des systémes Unix). Voir Section 6.1.3 [Name case sensitivity], page 408.
  - Les bases de donnèes, les tables, les index, les colonnes et les alias peuvent commencer par un chiffre (mais ne peuvent pas être constituès uniquement de chiffres).
  - Vous pouvez utiliser les commandes systémes standard pour sauver, renommer, dèplacer, effacer et copier des tables. Par exemple, pour renommer une table, il suffit de renommer les fichiers '`.MYD`', '`.MYI`' et '`.frm`' et de leur donner un nouveau nom.
- Dans une requête SQL, vous pouvez accèder á des tables situèes dans diffèrentes bases de donnèes, avec la syntaxe `db_name.tbl_name`. Certains serveurs SQL fournissent la même fonctionnalitè, mais l'appellent un `User space`. Le serveur MySQL ne supporte par les espaces de nom de tables, comme dans : `create table ralph.my_table...IN my_tablespace`.
- `LIKE` est possible avec des colonnes numèriques.
- Utilisez `INTO OUTFILE` et `STRAIGHT_JOIN` dans les requêtes `SELECT`. Voir Section 6.4.1 [SELECT], page 481.

- L'option `SQL_SMALL_RESULT` de la commande `SELECT`.
- La commande `EXPLAIN SELECT` pour avoir le dètail des jointures de tables.
- L'utilisation de noms d'index, de prèfixes d'index, et l'utilisation des mots-clès `INDEX` or `KEY` dans une commande de crèation de table `CREATE TABLE`. Voir Section 6.5.3 [`CREATE TABLE`], page 504.
- L'utilisation des clauses `TEMPORARY` et `IF NOT EXISTS` avec `CREATE TABLE`.
- L'utilisation de `COUNT(DISTINCT list)` oú `list` contient plus d'un èlèment.
- L'utilisation de `CHANGE col_name`, `DROP col_name` ou `DROP INDEX`, `IGNORE` ou `RENAME` dans une commande `ALTER TABLE`. Voir Section 6.5.4 [`ALTER TABLE`], page 512.
- L'utilisation de `RENAME TABLE`. Voir Section 6.5.5 [`RENAME TABLE`], page 516.
- L'utilisation de multiple `ADD`, `ALTER`, `DROP` et `CHANGE` dans les clauses de la commande `ALTER TABLE`.
- L'utilisation de `DROP TABLE` avec les mots-clès `IF EXISTS`.
- Vous pouvez effacer plusieurs tables avec une seule commande `DROP TABLE`.
- La clause `LIMIT` de la commande `DELETE`.
- La clause `DELAYED` des commandes `INSERT` et `REPLACE`.
- La clause `LOW_PRIORITY` des commandes `INSERT`, `REPLACE`, `DELETE` et `UPDATE`.
- L'utilisation de la commande `LOAD DATA INFILE`. Dans de nombreuses situations, cette syntaxe est compatible avec la commande d'Oracle `LOAD DATA INFILE`. Voir Section 6.4.9 [`LOAD DATA`], page 497.
- Les commandes `ANALYZE TABLE`, `CHECK TABLE`, `OPTIMIZE TABLE` et `REPAIR TABLE`.
- La commande `SHOW`. Voir Section 4.5.6 [`SHOW`], page 267.
- Les chaînes de caractéres peuvent être soit dèlimitèes par '`"`', soit par '`''`'. Pas seulement par '`''`'.
- L'utilisation du caractére de protection '`\`'.
- La commande `SET`. Voir Section 5.5.6 [`SET`], page 396.
- Vous n'êtes pas obligè de nommer toutes les colonnes que vous sèlectionnez dans la clause `GROUP BY`. Cela donne de meilleures performances pour certaines situations spècifiques, mais classiques. Voir Section 6.3.7 [Group by functions], page 479.
- Vous pouvez spècifier `ASC` ou `DESC` dans la clause `GROUP BY`.
- Pour aider les utilisateurs qui viennent d'autres environnements SQL, le serveur MySQL supporte des alias de nombreuses fonctions. Par exemple, toutes les fonctions de chaînes de caractéres supportent simultanèment les syntaxes ANSI SQL et ODBC.
- Le serveur MySQL comprend les opèrateurs `||` et `&&` comme opèrateurs logiques OR et AND, comme en langage C. Pour le serveur MySQL, les opèrateurs `||` et `OR` sont synonymes, ainsi que `&&` et `AND`. En consèquence, MySQL ne supporte pas l'opèrateur de concatènation de chaînes ANSI SQL `||`. Utilisez plutôt la fonction `CONCAT()`. Comme `CONCAT()` prend un nombre illimitè d'arguments, il est facile de convertir des expressions utilisant `||`, pour qu'elles fonctionnent sur le serveur MySQL.
- `CREATE DATABASE` et `DROP DATABASE`. Voir Section 6.5.1 [`CREATE DATABASE`], page 503.
- L'opèrateur `%` est synonyme de `MOD()`. C'est á dire que `N % M` est èquivalent á `MOD(N,M)`. `%` est supportè pour les programmeurs C, et pour la compatibilitè avec PostgreSQL.

- Les opèrateurs `=`, `<>`, `<=` ,`<`, `>=`,`>`, `<<`, `>>`, `<=>`, `AND`, `OR` ou `LIKE` peuvent être utilisès pour les comparaisons de colonnes á gauche de la clause `FROM` dans les commandes `SELECT`. Par exemple :

```
mysql> SELECT col1=1 AND col2=2 FROM tbl_name;
```

- La fonction `LAST_INSERT_ID()`. Voir Section 8.4.3.30 [`mysql_insert_id()`], page 635.
- Les opèrateurs d'expressions règuliéres ètendus `REGEXP` et `NOT REGEXP`.
- `CONCAT()` et `CHAR()` avec un argument ou plus de deux arguments. Avec le serveur MySQL, ces fonctions peuvent prendre n'importe quel nombre d'arguments.
- Les fonctions `BIT_COUNT()`, `CASE`, `ELT()`, `FROM_DAYS()`, `FORMAT()`, `IF()`, `PASSWORD()`, `ENCRYPT()`, `MD5()`, `ENCODE()`, `DECODE()`, `PERIOD_ADD()`, `PERIOD_DIFF()`, `TO_DAYS()` et `WEEKDAY()`.
- L'utilisation de la fonction `TRIM()` pour rèduire les chaînes. L'ANSI SQL ne supporte que les suppressions de caractéres uniques.
- Les fonctions de groupe de la clause `GROUP BY STD()`, `BIT_OR()` et `BIT_AND()`.
- L'utilisation de `REPLACE` á la place de `DELETE + INSERT`. Voir Section 6.4.8 [`REPLACE`], page 496.
- Les commandes `FLUSH`, `RESET` et `DO`.
- La possibilitè de modifier les variables dans les commandes avec l'opèrateur `:=` :

```
SELECT @a:=SUM(total),@b=COUNT(*),@a/@b AS avg FROM test_table;
SELECT @t1:=(@t2:=1)+@t3:=4,@t1,@t2,@t3;
```

1.7.4 Diffèrences de MySQL avec ANSI SQL92

Nous tâchons de rendre le serveur MySQL compatible avec le standard ANSI SQL, et le standard ODBC SQL, mais dans certains cas, MySQL se comporte diffèremment.

- Pour les colonnes de type `VARCHAR`, les espaces terminaux sont supprimès lors du stockage de la valeur. Voir Section 1.7.5 [Bugs], page 43.
- Dans certains cas, les colonnes `CHAR` sont transformèes automatiquement en colonnes `VARCHAR`. Voir Section 6.5.3.1 [Silent column changes], page 511.
- Les droits d'un utilisateur sur une table ne sont pas supprimès si la table est dètruite. Vous devez explicitement utiliser la commande `REVOKE` pour supprimer les droits d'un utilisateur sur une table. Voir Section 4.3.1 [`GRANT`], page 226.
- `NULL AND FALSE` donnera la valeur `NULL` et non pas `FALSE`. Nous pensons que dans cette situation, il n'est pas bon d'èvaluer de nombreuses conditions supplèmentaires.

Pour voir la liste des prioritès de dèveloppement des nouvelles extensions de MySQL, consultez la liste de tâches sur `http://www.mysql.com/doc/en/TODO.html`. C'est la toute derniére liste de tâche de ce manuel. Voir Section 1.8 [TODO], page 46.

1.7.4.1 Sous sèlections (Sub`SELECT`s)

Jusqu'á la version 4.0, le serveur MySQL ne prenait en charge que les requêtes imbriquèes de la forme `INSERT ... SELECT ...` et `REPLACE ... SELECT ...`. Vous pouvez nèanmoins

utiliser la fonction `IN()` dans d'autres contextes. Les sous-requêtes sont implèmentèes dans l'arborescence de dèveloppement 4.1.

En attendant, vous pourrez dans la plupart des cas rèècrire la requête sans sous-requêtes :

```
SELECT * FROM table1 WHERE id IN (SELECT id FROM table2);
```

Ceci peut être rèècrit de la façon suivante :

```
SELECT table1.* FROM table1,table2 WHERE table1.id=table2.id;
```

Les requêtes :

```
SELECT * FROM table1 WHERE id NOT IN (SELECT id FROM table2);
SELECT * FROM table1 WHERE NOT EXISTS (SELECT id FROM table2
                                       WHERE table1.id=table2.id);
```

peuvent être rèècrites ainsi :

```
SELECT table1.* FROM table1 LEFT JOIN table2 ON table1.id=table2.id
                                       WHERE table2.id IS NULL;
```

Pour des requêtes plus complexes, vous pouvez gènèralement crèer des tables temporaires contenant les sous-requêtes. Cependant, dans certains cas cette option ne fonctionnera pas. Le cas le plus frèquent se produit avec les instructions `DELETE`, pour lesquelles le SQL standard ne prend pas en charge les jointures (sauf dans les sous Select). Dans une telle situation, en attendant que les sous-requêtes soient prises en charge par MySQL Server, vous disposez de deux options.

La premiére option consiste á utiliser un langage de programmation procèdural (tel que Perl ou PHP) afin de soumettre une requête `SELECT` pour obtenir les clès primaires des enregistrements á supprimer. On utilise ensuite ces valeurs pour construire l'instruction `DELETE` (`DELETE FROM ... WHERE ... IN (key1,key2, ...)`).

La seconde option consiste á utiliser le SQL interactif pour construire automatiquement un ensemble d'instructions `DELETE`, grâce á l'extension MySQL `CONCAT()` (á la place de l'opèrateur standard `||`).

Par exemple :

```
SELECT CONCAT('DELETE FROM tab1 WHERE pkid = ', "'", tab1.pkid, "'", ';')
  FROM tab1, tab2
 WHERE tab1.col1 = tab2.col2;
```

Vous pouvez placer cette requête dans un fichier script et rediriger son entrèe vers l'interprèteur de ligne de commande `mysql`, tout en ouvrant un canal de communication (pipe) pour diriger sa sortie vers une seconde instance de l'interprèteur :

```
shell> mysql --skip-column-names mydb < myscript.sql | mysql mydb
```

La version 4.0 de MySQL prend en charge les instructions `DELETE` sur plusieurs tables pouvant être utilisèes afin de supprimer efficacement des lignes á partir d'informations issues d'une table, voire de plusieurs tables en même temps.

1.7.4.2 SELECT INTO TABLE

Le serveur MySQL ne supporte pas encore l'extension Oracle SQL : `SELECT ... INTO TABLE ...`. A la place, le serveur MySQL supporte la syntaxe ANSI SQL `INSERT INTO ... SELECT ...`, qui revient au même. Voir Section 6.4.3.1 [INSERT SELECT], page 491.

```
INSERT INTO tblTemp2 (fldID) SELECT tblTemp1.fldOrder_ID
       FROM tblTemp1 WHERE tblTemp1.fldOrder_ID > 100;
```

Vous pouvez aussi utiliser `SELECT INTO OUTFILE...` ou `CREATE TABLE ... SELECT`.

1.7.4.3 Transactions et opèrations atomiques

Le serveur MySQL support les transactions avec les gestionnaires de tables `InnoDB` et `BDB`. Voir Chapitre 7 [Table types], page 531. `InnoDB` dispose aussi de la compatibilitè `ACID`.

Toutefois, les tables non transactionnelles de MySQL telles que `MyISAM` exploitent un autre concept pour assurer l'intègritè des donnèes, appelè "`opèrations atomiques`". Les opèrations atomiques disposent d'une bien meilleure protection des donnèes pour des performances ègalement accrues. Comme MySQL supporte les deux mèthodes, l'utilisateur est capable de choisir celle qui correspond á ses besoins, suivant qu'il a besoin de vitesse ou de sècuritè. Ce choix peut être fait table par table.

Comment exploiter les capacitès de MySQL pour protèger l'intègritè des donnèes, et comment ces fonctionnalitès se comparent-elles avec les mèthodes transactionnelles ?

1. En mode transactionnel, si votre application a ètè ècrite en dèpendant de l'appel de `ROLLBACK` au lieu de `COMMIT` dans les situations critiques, les transactions sont plus plus pratiques. Les transactions s'assurent que les modifications non achevèes ou les activitès corrosives ne sont pas archivèes dans la base. Le serveur a l'opportunitè d'annuler automatiquement l'opèration, et votre base de donnèes est sauve.

   Le serveur MySQL, dans la plupart des cas, vous permet de rèsoudre les problémes potentiels en incluant de simples vèrifications avant les modifications, et en exècutant des scripts simples pour vèrifier l'intègritè de vos bases de donnèes, ainsi que les incohèrences, et pour rèparer automatiquement les problémes, ou encore vous alerter si une erreur est identifièe. Notez qu'en utilisant simplement le log de MySQL, ou en utilisant un log supplèmentaire, vous pouvez normalement rèparer á la perfection toutes les tables, sans aucune perte de donnèes.
2. Souvent, les modifications de donnèes transactionnelles fatales peuvent être rèècrites de maniére atomique. En gènèral, tous les problémes d'intègritè que les transactions rèsolvent peuvent être corrigès avec la commande `LOCK TABLES` ou des modifications atomiques, qui assurent que vous n'aurez jamais d'annulation automatique de la base, ce qui est un probléme commun des bases transactionnelles.
3. Même un systéme transactionnel peut perdre des donnèes si le serveur s'arrête. La diffèrence entre les systémes repose alors dans ce petit laps de temps oú ils peuvent perdre des donnèes. Aucun systéme n'est sècurisè á 100 %, mais simplement "suffisament sècurisè". Même Oracle, rèputè pour être la plus sûre des bases de donnèes transactionnelles, est montrè du doigt pour perdre des donnèes dans ces situations.

   Pour être tranquille avec MySQL, que vous utilisiez les tables transactionnelles ou pas, vous n'avez besoin que de sauvegardes et de logs de modifications. Avec ces deux outils, vous pourrez vous protèger de toutes les situations que vous pourriez rencontrer avec d'autres bases de donnèes transactionnelles. De toute maniére, il est bon d'avoir des sauvegardes, indèpendamment de la base que vous utilisez.

La mèthode transactionnelle a ses avantages et ses inconvènients. De nombreux utilisateurs et dèveloppeurs d'applications dèpendent de la facilitè de pallier un probléme lorsqu'une

annulation semble nècessaire ou presque. Cependant, même si vous êtes nèophyte des opèrations atomiques, ou plus familier avec les transactions, prenez en considèration le gain de vitesse que les tables non transactionnelles offrent. Ces gains vont de 3 a 5 fois la vitesse des tables transactionnelles les plus rapides et les mieux optimisèes.

Dans des situations oú l'intègritè est de la plus grande importance, le serveur MySQL assure une intègritè du niveau des transactions, ou encore mieux avec les tables non transactionnelles. Si vous verrouillez les tables avec `LOCK TABLES`, toutes les modifications seront bloquèes jusqu'á ce que la vèrification d'intègritè soit faite (á comparer avec un verrou en ècriture), les lectures et insertions sont toujours possibles. Les nouvelles lignes ne seront pas accessibles en lecture tant que le verrou n'aura pas ètè levè. Avec `INSERT DELAYED`, vous pouvez faire attendre les insertions dans une pile, jusqu'á ce que les verrous soit levès, sans que le client n'attende cette levèe de verrou. Voir Section 6.4.4 [INSERT DELAYED], page 491.

"Atomique", avec le sens que nous lui donnons, n'a rien de magique. Ce terme signifie simplement que vous pouvez être certain que lorsque vous modifiez des donnèes dans une table, aucun autre utilisateur ne peut interfèrer avec votre opèration, et qu'il n'y aura pas d'annulation automatique (ce qui pourrait arriver avec des tables transactionnelles si nous ne sommes pas trop soigneux). Le serveur MySQL garantit aussi qu'il n'y aura pas de lectures erronèes.

Voici quelques techniques pour travailler avec des tables non transactionnelles :

- Les boucles qui requiérent les transactions peuvent normalement être implèmentèes avec la commande `LOCK TABLES`, et vous n'avez nul besoin de curseur lorsque vous modifiez des lignes á la volèe.
- Pour èviter d'utiliser l'annulation `ROLLBACK`, vous pouvez adopter la stratègie suivante :
  1. Utilisez la commande `LOCK TABLES ...` pour verrouiller toutes les tables que vous voulez utiliser.
  2. Testez vos conditions.
  3. Modifiez si tout est correct.
  4. Utilisez `UNLOCK TABLES` pour libèrer vos tables.

  Ceci est probablement une mèthode bien plus rapide que ne le proposent les transactions, avec des annulations `ROLLBACK` possibles mais pas certaines. La seule situation que ce cas ne prend pas en compte est l'interruption du processus au milieu d'une mise á jour. Dans ce cas, tous les verrous seront levès, mais certaines modifications peuvent ne pas avoir ètè exècutèes.
- Vous pouvez aussi utiliser des fonctions pour modifier des lignes en une seule opèration. Vous pouvez crèer une application trés efficace en utilisant cette technique :
  - Modifiez les champs par rapport á leur valeur actuelle.
  - Modifiez uniquement les champs que vous avez rèellement changè.

  Par exemple, lorsque nous modifions les donnèes d'un client, nous ne modifions que les donnèes du client qui ont changè et nous vèrifions uniquement si les donnèes modifièes ou les donnèes qui en dèpendent ont changè comparativement aux donnèes originales. Les tests sur les donnèes modifièes sont faits avec la clause `WHERE` dans la commande

UPDATE. Si la ligne a ètè modifièe, nous indiquons au client : "Some of the data you have changed has been changed by another user". En français : "certaines donnèes que vous voulez modifier ont ètè modifièes par un autre utilisateur". Puis nous affichons l'ancienne ligne et la nouvelle ligne, pour laisser l'utilisateur dècider quelle version il veut utiliser.

Cela nous conduit á un rèsultat proche du verrouillage de ligne, mais en fait, c'est bien mieux, car nous ne modifions que les colonnes qui en ont besoin, en utilisant des valeurs relatives. Cela signifie qu'une commande UPDATE typique ressemble á ceci :

```
UPDATE tablename SET pay_back=pay_back+'relative change';

UPDATE customer
  SET
    customer_date='current_date',
    address='new address',
    phone='new phone',
    dette=dette+'emprunt'
  WHERE
    customer_id=id AND address='old address' AND phone='old phone';
```

Comme vous pouvez le voir, c'est trés efficace, et fonctionne même si un autre client a modifiè la valeur pay_back ou dette.

- Dans de nombreuses situations, les utilisateurs ont souhaitè les commandes ROLLBACK et/ou LOCK TABLES afin de gèrer des identifiant uniques pour certaines tables. Ils peuvent être gèrès bien plus efficacement en utilisant une colonne de type AUTO_INCREMENT, en corrèlation avec la fonction LAST_INSERT_ID() ou la fonction C mysql_insert_id(). Voir Section 8.4.3.30 [mysql_insert_id()], page 635.

  Vous pouvez èviter le verrouillage de ligne. Certaines situations le requiérent vraiment, mais elles sont rares. Les tables InnoDB supportent le verrouillage de ligne. Avec les tables MyISAM, vous pouvez utiliser une colonne de type flag, et faire ceci :

  ```
  UPDATE tbl_name SET row_flag=1 WHERE id=ID;
  ```

  MySQL retournera 1 pour le nombre de lignes affectèes si la ligne a ètè trouvèe, car row_flag ne vaut pas dèjá 1 dans la ligne originale.

  Vous pouvez comprendre la requête ci-dessus comme si le serveur MySQL avait utilisè la commande suivante :

  ```
  UPDATE tbl_name SET row_flag=1 WHERE id=ID AND row_flag <> 1;
  ```

1.7.4.4 Procèdures stockèes et triggers

Une procèdure stockèe est une liste de commandes SQL qui peuvent être compilèes et stockèes sur le serveurs. Une fois que cela est fait, les clients n'ont pas besoin de soumettre á nouveau toute la commande, mais font simplement rèfèrence á la procèdure stockèe. Cela conduit á des performances bien meilleures, car les commandes n'ont pas á être analysèes plusieurs fois, et que bien moins d'informations transitent sur le rèseau. Vous pouvez aussi èlevez le niveau conceptuel de votre application en mettant en place des librairies de fonctions sur le serveur.

Un trigger est une procèdure stockèe qui est activèe lorsqu'un èvènement particulier survient. Par exemple, vous pouvez installer une procèdure stockèe qui est dèclenchèe dés qu'une ligne est effacèe dans une table d'achat, pour que le client soit automatiquement effacè si tous ses achats sont effacès.

La future version du langage sera capable de supporter les procèdures stockèes. Notre objectif est d'impolèmenter les procèdures stockèes pour la version 5.0 du serveur MySQL. Nous nous pencherons aussi sur les triggers.

1.7.4.5 Les clès ètrangéres

Notez que les clès en SQL ne sont pas utilisèes pour joindre des tables, mais pour assurer l'intègritè rèfèrentielle (contraintes de clès ètrangéres). Si vous voulez obtenir des rèsultats issus de tables multiples avec une commande `SELECT`, vous allez joindre les tables comme ceci :

```
SELECT * FROM table1,table2 WHERE table1.id = table2.id;
```

Voir Section 6.4.1.1 [JOIN], page 486. Voir Section 3.5.6 [example-Foreign keys], page 182.

En MySQL version 3.23.44 et plus rècentes, les tables `InnoDB` supportent les vèrifications d'intègritè rèfèrentielles. Voir Section 7.5 [InnoDB], page 544. Pour les autres types de tables, le serveur mySQL accepte la syntaxe `FOREIGN KEY` dans la commande `CREATE TABLE`, mais ne la prend pas en compte.

La syntaxe `FOREIGN KEY` dans la clause `ON DELETE ...` est principalement utilisèe pour des raisons de documentation. Certaines applications ODBC peuvent utiliser cette clause pour produire des conditions `WHERE` automatiques, mais c'est gènèralement simple á èviter. `FOREIGN KEY` est parfois utilisè pour vèrifier une contrainte, mais cette vèrification n'est pas nècessaire en pratique, si les lignes sont insèrèes dans la table dans le bon ordre.

Avec le serveur MySQL, vous pouvez contourner le probléme de l'absence de la clause `ON DELETE ...` en ajoutant la commande `DELETE` appropriée dans l'application, lorsque vous effacez une ligne dans une table qui a une clè ètrangére. En pratique, cette technique est aussi rapide (et même parfois plus rapide), et bien plus portable que l'utilisation des clès ètrangéres.

Avec le serveur MySQL 4.0, vous pouvez utiliser les commandes d'effacement multi-tables pour effacer des lignes dans plusieurs tables en une seule commande. Voir Section 6.4.6 [DELETE], page 494.

Dans un futur proche, nous allons ètendre l'implèmentation de la clause `FOREIGN KEY` pour que cette information soit stockèe dans le fichier de spècification des tables et puisse être lu par `mysqldump` et ODBC. Dans une ètape ultèrieure, nous allons implèmenter les contraintes de clè ètrangére pour les applications qui ne peuvent pas être codées sans.

Gardez bien en tête que les clès ètrangéres sont souvent mèconnues, ce qui peut causer de graves problémes. Même lorsqu'elles sont utilisèes correctement, ce n'est pas une solution magique pour les problémes d'intègritè rèfèrentielle, même si cela simplifie parfois les choses.

Voici des avantages aux contraintes de clès ètrangéres :

- En supposant que les relations soient proprement conçues, les clès ètrangéres rendent plus difficile pour un programmeur d'insèrer des valeurs incohèrentes dans la base.
- L'utilisation des modifications et effacement en cascade simplifie le code du client.

- Les régles de clès ètrangéres proprement conçues aident á la documentation des relations entre les tables.

Inconvènients :

- Les erreurs, qui sont faciles á faire durant la conception des relations, peuvent causer des problémes graves : par exemple, des régles de contrainte circulaires, ou des combinaisons erronèes d'effacement.
- Une application correctement ècrite s'assure d'elle-même de l'intègritè rèfèrentielle des donnèes avant d'exècuter une requête. De ce fait, les vèrifications supplèmentaires de la base sont inutiles et rèduisent les performances de l'application.
- Il n'est pas rare pour un administrateur de bases de donnèes de faire une topologie complexe des relations entre tables qui rendent trés complexe, voire impossible de restaurer des tables.

1.7.4.6 Les vues

Il est prèvu d'implèmenter les vues dans la version 5.0 du serveur MySQL.

Les vues sont la plupart du temps utiles pour donner accés aux utilisateurs á un ensemble de relations repròsentèes par une table (en mode inaltèrable). Beaucoup de bases de donnèes SQL ne permettent pas de mettre á jour les lignes dans une vue, vous devez alors faire les mises á jour dans les tables sèparèes.

Vu que le serveur MySQL est la plupart du temps utilisè dans des applications oú le dèveloppeur á un contrôle total sur l'utilisation de la base de donnèes, la plupart de nos utilisateurs n'ont pas considèrè les vues comme trés importantes (du moins, personne n'a ètè assez intèressè á ce sujet pour être prêt á financer l'implèmentation des vues).

On n'a pas forcèment besoin des vues pour restreindre l'accés aux colonnes, et ce, parce que MySQL a un systéme de droits trés sophistiquè. Voir Section 4.2 [Privilege system], page 203.

1.7.4.7 '--' comme dèbut de commentaire

Certaines bases de donnèes SQL utilisent '`--`' comme dèbut de commentaire. Le serveur MySQL utilise '`#`'. Vous pouvez aussi utiliser la syntaxe du langage C `/* ceci est un commentaire */`. Voir Section 6.1.6 [Comments], page 413.

MySQL version 3.23.3 et plus rècent supporte les commentaires de type '`--`', si ce commentaire est suivi d'un espace. Ceci est dû au fait que ce type de commentaire a causè beaucoup de problémes avec les requêtes gènèrèes automatiquement, qui contiennent du code tel que celui-ci, oú nous insèrons automatiquement la valeur du paiement dans la table `!payment!` :

```
UPDATE tbl_name SET credit=credit-!payment!
```

Pensez á ce qui se passe lorsque la valeur de `payment` est nègative. Comme `1--1` est valide en SQL, la consèquence est de commencer un commentaire '`--`'.

En utilisant notre implèmentation des commentaires avec le serveur MySQL version 3.23.3 et plus rècent, `1-- ceci est un commentaire` ne pose pas ce type de probléme.

Une autre fonctionnalitè supplèmentaire est que le client en ligne de commande `mysql` supprime toutes les lignes qui commencent par '`--`'.

Les informations suivantes sont destinèes aux utilisateurs de MySQL avec des versions antèrieures á la version 3.23.3 :

Si vous avez un programme SQL dans un fichier texte qui contient des commentaires au format '`--`', il est recommandè d'utiliser :

```
shell> replace " --" " #" < text-file-with-funny-comments.sql \
         | mysql database
```

au lieu du classique :

```
shell> mysql database < text-file-with-funny-comments.sql
```

Vous pouvez aussi èditer le fichier de commande "lui-même" pour remplacer les commentaires '`--`' par des commentaires '`#`' :

```
shell> replace " --" " #" -- text-file-with-funny-comments.sql
```

Puis, rètablissez-les avec :

```
shell> replace " #" " --" -- text-file-with-funny-comments.sql
```

1.7.5 Erreurs connues et problémes de conceptions de MySQL

Les problémes suivants sont connus, et sont en tête de liste pour être corrigès :

- `ANALYZE TABLE`, sur une table de type BDB, peut rendre la table inutilisable, dans certains cas, jusqu'au prochain redèmarrage de `mysqld`. Lorsque cela survient, vous rencontrez les erreurs suivantes dans le fichier d'erreur MySQL :

  ```
  001207 22:07:56  bdb:  log_flush: LSN past current end-of-log
  ```

- N'exècutez pas de commande `ALTER TABLE` sur une table `BDB` sur laquelle vous avez exècutè des transactions á plusieurs commandes, jusqu'á ce que ces transactions soient achevèes : la transaction sera probablement ignorèe.
- `ANALYZE TABLE`, `OPTIMIZE TABLE` et `REPAIR TABLE` peuvent causer des problémes sur les tables avec lesquelles vous utilisez la commande `INSERT DELAYED`.
- Faire un `LOCK TABLE ...` et `FLUSH TABLES ...` ne vous garantit pas qu'il n'y a pas une transaction en court sur la table.
- Les tables BDB sont lentes á ouvrir. Si vous avez de nombreuses tables BDB dans une base, cela prendra du temps au client `mysql` pour accèder á la base si vous n'utilisez pas l'option `-A`, ou si vous utilisez la commande `rehash`. C'est particuliérement vrai si vous n'avez pas de cache de table important.

Les problémes suivants sont connus et seront corrigès en leur temps :

- Lorsque vous utilisez la commande `SET CHARACTER SET`, il n'est pas possible d'utiliser les caractéres traduits dans les noms de bases, de tables ou de colonnes.
- Il n'est pas possible d'utiliser `_` ou `%` avec la commande `ESCAPE` dans la clause `LIKE... ESCAPE`.
- Si vous avez une colonne de type `DECIMAL` avec un nombre stockè dans un autre format (+01.00, 1.00, 01.00), `GROUP BY` peut considèrer ces valeurs comme diffèrentes.

- Lorsque `DELETE FROM merge_table` est utilisè sans la clause `WHERE`, elle va simplement effacer le fichier de la table, et ne pas effacer les tables associèes.
- Vous ne pouvez pas compiler le serveur dans un autre dossier lorsque vous utilisez les MIT-pthreads. Comme cela requiert une modification des MIT-pthreads, nous ne corrigerons pas ce probléme. Voir Section 2.3.6 [MIT-pthreads], page 95.
- Les valeurs de type `BLOB` ne peuvent pas être utilisèes "correctement" dans les clauses `GROUP BY` ou `ORDER BY` ou `DISTINCT`. Seuls, les `max_sort_length` premiers octets (par dèfaut, 1024) seront utilisès pour les comparaisons de `BLOB`. Ceci peut être modifiè avec l'option `-O max_sort_length` de `mysqld`. Un palliatif á ce probléme est d'utiliser une sous partie de chaîne : `SELECT DISTINCT LEFT(blob,2048) FROM tbl_name`.
- Les calculs sont faits avec des `BIGINT` ou `DOUBLE` (les deux sont normalement de 64 bits). La prècision dèpend alors de la fonction utilisèe. La régle gènèrale est que les fonctions de bits utilisent la prècision des `BIGINT`, `IF` et `ELT()` utilisent la prècision des `BIGINT` ou `DOUBLE`, et les autres utilisent la prècision des `DOUBLE`. Il faut donc èviter d'utiliser les entiers non signès de grande taille, surtout s'ils dèpassent la taille de 63 bits (9223372036854775807) pour toute autre fonction que les champs de bits ! La version 4.0 gére bien mieux les `BIGINT` que la 3.23.
- Toutes les colonnes de type chaînes, hormis les `BLOB` et `TEXT`, voient automatiquement leurs caractéres blancs finaux supprimès. Pour le type `CHAR` c'est correct, et c'est considèrè comme une fonctionnalitè par la norme ANSI SQL92. Le hic est que pour le serveur MySQL les colonnes `VARCHAR` sont traitèes de la même façon.
- Vous ne pouvez avoir que des colonnes de taille 255 pour les `ENUM` et `SET`.
- Avec les fonctions d'agrègation `MIN()`, `MAX()` et compagnie, MySQL compare actuellement les colonnes de type `ENUM` et `SET` par leur valeur de chaîne, plutôt que par leur position relative dans l'ensemble.
- `safe_mysqld` redirige tous les messages de `mysqld` vers le log `mysqld`. Le probléme est que si vous exècutez `mysqladmin refresh` pour fermer et ouvrir á nouveau l'historique, `stdout` et `stderr` sont toujours redirigès vers l'ancien log. Si vous utilisez `--log` extensivement, vous devriez èditer `safe_mysqld` pour loger vers '`'hostname'.err`' au lieu de '`'hostname'.log`', de façon á pouvoir facilement rècupèrer la place de l'ancien log, en effaçant les vieux, et en exècutant `mysqladmin refresh`.
- Dans la commande `UPDATE`, les colonnes sont modifièes de gauche á droite. Si vous faite rèfèrence á une colonne modifièe, vous obtiendrez sa valeur modifièe, plutôt que sa valeur originale. Par exemple :

  ```
  mysql> UPDATE tbl_name SET KEY=KEY+1,KEY=KEY+1;
  ```

  Cette commande va modifier la colonne `KEY` avec `2` au lieu de `1`.
- Vous ne pouvez pas utiliser les tables temporaires plus d'une fois dans la même requête. Par exemple, cette commande ne fonctionne pas :

  ```
  mysql> SELECT * FROM temporary_table, temporary_table AS t2;
  ```
- `RENAME` ne fonctionne pas avec les tables `TEMPORARY`, ou les tables utilisèes dans un rassemblement (`MERGE`).
- L'optimiseur peut gèrer la clause `DISTINCT` diffèremment si vous utilisez des colonnes cachèes dans une jointure. Dans une jointure, les colonnes cachèes sont comptèes comme une partie du rèsultat (même si elles ne sont pas montrèes), tandis que dans les

requêtes normales, les colonnes cachèes ne participent pas aux `DISTINCT`. Nous allons probablement modifier ceci dans le futur, pour ne jamais exploiter les colonnes cachèes avec `DISTINCT`.

Voici un exemple :

```
SELECT DISTINCT mp3id FROM band_downloads
       WHERE userid = 9 ORDER BY id DESC;
```

et

```
SELECT DISTINCT band_downloads.mp3id
       FROM band_downloads,band_mp3
       WHERE band_downloads.userid = 9
       AND band_mp3.id = band_downloads.mp3id
       ORDER BY band_downloads.id DESC;
```

Dans le second cas, MySQL 3.23.x pourrait vous donner deux lignes identiques dans le rèsultat (car les lignes cachèes `id` différent).

Notez que cela n'arrive que pour les requêtes oú vous n'avez pas de colonnes de la clause ORDER BY dans le rèsultat, ce que vous ne pourriez pas faire en ANSI SQL.

- Comme le serveur MySQL vous permet de travailler avec des tables qui ne supportent pas les transactions, et donc, l'annulation `rollback`, certains comportements sont diffèrents avec MySQL d'avec d'autres serveurs SQL. C'est nècessaire pour s'assurer que MySQL n'a jamais besoin d'annuler une commande SQL. Cela peut sembler un peu ètrange au moment ou les colonnes doivent être vèrifièes par l'application, mais cela vous fournit une accèlèration notable, á cause d'optimisations qui ne pourraient pas avoir lieu ailleurs.

  Si vous donnez une valeur incorrecte á une colonne, MySQL va stocker le `meilleur code possible` dans la colonne, au lieu d'annuler la transaction :

  - Si vous essayez de stocker une valeur qui est hors de l'intervalle de validitè dans une colonne numèrique, MySQL va stocker la plus petite ou la plus grande valeur qu'il connaisse dans cette colonne.
  - Si vous essayez de stocker une chaîne qui ne commence pas pas un chiffre dans une colonne numèrique, MySQL va stocker 0.
  - Si vous essayez de stocker la valeur `NULL` dans une colonne qui n'accepte pas la valeur `NULL`, le serveur MySQL va stocker 0 ou '' (chaîne vide) á la place : ce comportement peut être modifiè avec l'option de compilation -DDONT_USE_DEFAULT_FIELDS).
  - MySQL vous autorise le stockage de dates erronèes dans les colonnes de type `DATE` et `DATETIME` (comme 2000-02-31 ou 2000-02-00). L'idèe est que ce n'est pas au serveur SQL de faire le travail de validation. Si MySQL peut stocker une date, et relire exactement cette date, alors MySQL va stocker cette date. Si la date est totalement fausse (hors de l'intervalle de validitè du serveur), la valeur spèciale 0000-00-00 sera utilisèe.
  - Si vous utilisez une valeur non supportèe avec une colonne de type `ENUM`, la valeur stockèe sera la chaîne vide, de valeur numèrique 0.
  - Si vous utilisez une valeur invalide dans une colonne de type `SET`, la valeur sera ignorèe.

- Si vous exècutez une `PROCEDURE` sur une requête qui retourne un rèsultat vide, dans certains cas, `PROCEDURE` ne transformera pas les colonnes.
- La crèation de table de type `MERGE` ne vèrifie pas si les tables sous-jacentes sont de type compatible.
- Le serveur MySQL ne supporte pas encore les valeurs Server `NaN`, `-Inf` et `Inf` pour les doubles. Utiliser ces valeurs gènèrera des problémes lorsque vous essayerez d'exporter et d'importer des donnèes. Comme solution temporaire, vous pouvez remplacer `NaN` par `NULL` (si possible) et `-Inf` et `Inf` par les valeurs maximales possibles des colonnes `double`.
- Les valeurs nègatives passèes á `LIMIT` sont traitèes comme des valeurs positives.
- Si vous utilisez la commande `ALTER TABLE` pour ajouter un index de type `UNIQUE` á un table utilisèe dans un rassemblement de tables `MERGE`, puis que vous utilisez `ALTER TABLE` pour ajouter un index normal á la table `MERGE`, l'ordre des clès sera diffèrent pour les tables s'il y avait dèjá une ancienne clè qui n'ètait pas unique. Ceci est dû au fait que `ALTER TABLE` place les clès `UNIQUE` avant les clès normales, pour être capable de dètecter les clès doublons plus vite.

Les bogues suivants sont connus dans les anciennes versions de MySQL :

- Vous pouvez obtenir un thread gelè si vous utilisez la commande `DROP TABLE` sur une table qui fait partie des tables verrouillèes par `LOCK TABLES`.
- Dans les cas suivants, vous pouvez obtenir un crash :
  - Le gestionnaire d'insertions retardèes a dèjá des insertions en attente pour une table.
  - `LOCK table` avec `WRITE`.
  - `FLUSH TABLES`.
- Pour les versions de MySQL avant la 3.23.2, une commande `UPDATE` qui modifiait une clè avec la clause `WHERE` sur la même clè, pouvait èchouer car la même clè ètait utilisèe pour rechercher les lignes et la même ligne pouvait être trouvèe plusieurs fois :

  ```
  UPDATE tbl_name SET KEY=KEY+1 WHERE KEY > 100;
  ```

  Un palliatif est :

  ```
  mysql> UPDATE tbl_name SET KEY=KEY+1 WHERE KEY+0 > 100;
  ```

  Cela fonctionnera, car MySQL ne va pas utiliser d'index sur une expression dans la clause `WHERE`.
- Avant la version 3.23 de MySQL, tous les types numèriques ètaient traitès comme des champs á virgule fixe. Cela signifie que vous deviez spècifier le nombre de dècimales que le champ devait avoir. Tous les rèsultats ètaient retournès avec le nombre correct de dècimales.

Pour les bogues spècifiques aux systémes d'exploitation, voyez la section sur la compilation et le port.

1.8 Les èvolutions de MySQL (la liste des tâches)

Cette section liste les fonctionnalitès que nous prèvoyons d'ajouter á MySQL.

La liste est grossiérement rèalisèe dans l'ordre de prioritè. Si vous voulez modifier cet ordre, achetez une licence de support, et dites-nous ce que vous souhaitez voir en premier. Voir Section 1.4 [Licensing and Support], page 16.

L'objectif est de supporter complétement la norme ANSI SQL99, tout en lui ajoutant de nombreuses amèliorations. Le dèfi est de faire tout cela sans sacrifier la vitesse d'exècution, ni la qualitè du code.

1.8.1 Ce que devrait inclure la version 4.0

Tout est fait. Nous ne fournissons plus que des corrections de bogues pour MySQL 4.0. Voir Section D.2 [News-4.0.x], page 728. Le dèveloppement est passè á la version 4.1

1.8.2 Ce qui est prèvu pour la version 4.1

Les fonctionnalitès suivantes sont prèvues pour être ajoutèes dés la version 4.1. Notez que comme nous avons de nombreux dèveloppeurs qui travaillent sur diffèrents projets, il y aura aussi de nombreuses amèliorations. Il y a aussi de faibles chances pour que ces fonctionnalitès soient ajoutèes á la version 4.0. Certaines fonctionnalitès sont dèjá installèes dans la version MySQL 4.1.

- Sous-sèlections.

  ```
  SELECT id FROM t WHERE grp IN (SELECT grp FROM g WHERE u > 100);
  ```

- Un nouveau format de dèfinition de table (fichiers '`.frm`'). Il nous permettra de ne pas être á court de place lors de l'ajout de nouvelles options. Les anciens fichiers sont toujours compatibles avec la version 4.0. Toutes les nouvelles tables utiliseront ce nouveau format.

  Le nouveau format de fichier nous permettra d'ajouter de nouveaux types de colonnes, plus d'options pour les clès, et alloueront de la place pour les dèfinitions des `FOREIGN KEY`.

- `SHOW COLUMNS FROM table_name` (utilisè par le client `mysql` pour afficher le dètail des colonnes) ne devrait pas ouvrir la table, mais uniquement le fichier de dèfinition. Cela consommera moins de mèmoire, et sera plus rapide.

- Le support des clès ètrangéres pour les tables `MyISAM`, incluant les effacement en cascade.

- La rèplication infaillible.

- La rèplication devrait fonctionner avec la fonction `RAND()` et la variable utilisateur `@var`.

- Sauvegarde á chaud, avec faible rèduction des performances. La sauvegarde en ligne rendra simple le nouveau systéme de rèplication, sans èteindre le serveur maître.

- Tables dèrivèes :

  ```
  SELECT a.col1, b.col2
         FROM (SELECT MAX(col1) AS col1 FROM root_table) a,
         other_table b
         WHERE a.col1=b.col1;
  ```

Cela peut se faire automatiquement, en crèant des tables temporaires pour les tables dèrivèes, pour la durèe de la requête.

- Fonction de groupement `ROLLUP` et `CUBE` OLAP (Online Analytical Processing) pour les applications de dèpôts de donnèes.
- Permettre que la commande `DELETE`, sur les tables `MyISAM`, utilise le cache de lignes. Pour cela, nous devons modifier les threads de cache de ligne lorsque nous modifions le fichier '`.MYD`'.
- Lors de l'utilisation de `SET CHARACTER SET`, il faudrait traduire toute la requête, et non pas seulement les chaînes. Cela permettra aux utilisateurs d'utiliser les caractéres traduits dans les noms de tables, bases et colonnes.
- Ajouter la mèthode `record_in_range()` á la commande `MERGE` pour être capable de choisir le bon index lorsqu'il y a trop de choix. Nous devrions aussi ètendre l'interface d'informations pour lire la distribution des clès de chaque index, si `analyze` est exècutè sur toutes les tables.
- Rèsoudre le probléme avec la commande `RENAME TABLE` lorsqu'elle est utilisèe sur une table active `MERGE`, ce qui conduit actuellement á la corruption de la table.
- Une librairie MySQL intègrèe plus rapide est plus compacte (compatible avec l'ancienne).
- Support OpenSSL stable (MySQL 4.0 supporte OpenSSL mais cela reste rudimentaire et pas totalement testè).
- Ajout du support en fonction de l'`UNICODE`.
- Transtypage de jeux de caractéres et des syntaxes pour gèrer simultanèment plusieurs jeux de caractéres.
- Aide pour toutes les commandes du client.
- Nouveau protocole client/serveur plus rapide qui supportera les commandes prèparèes, les paramétres liès, et les rèsultats liès, les transferts binaires de donnèes, les alertes...
- Ajout des vrais noms de tables et de bases (dans le cas d'un alias) á la structure MYSQL_FIELD.
- Ajout d'options au protocole client/serveur pour obtenir des informations sur la progression des requêtes lentes.
- L'implèmentation de `RENAME DATABASE`. Pour que cela soit sècuritaire pour tous les gestionnaires de table, cela doit fonctionner comme ceci :
  - Crèer la nouvelle base de donnèes.
  - Pour chaque table, renommer la table dans la nouvelle base de donnèes, en utilisant l'èquivalent de la commande `RENAME`.
  - Effacer l'ancienne base de donnèes.
- Ajouter le vrai support des `VARCHAR` (Il y a dèjá ce support avec les tables `MyISAM`).
- Optimiser le type `BIT` pour qu'il prenne un bit (actuellement, `BIT` prend un caractére).
- Nouvelle interface interne avec les fichiers. Cela rendra plus gènèrique la gestion des fichiers, et rendra plus simple des extensions du type RAID. (L'implèmentation actuelle est un hack.)
- Amèliorer les tables en mèmoire (`HEAP`) :

- Support des index B-tree
- Lignes de taille dynamique
- Accèlèration de la gestion des lignes (moins de copie)

1.8.3 Ce qui doit être fait dans un futur proche

- Limiter le nombre de threads qui travaillent á la restauration d'une table MyISAM en même temps.
- Changer le code de la commande `INSERT ... SELECT` pour supporter les insertions concurrentes, en option.
- Retourner le type original des champs avec la commande `SELECT MIN(column) ... GROUP BY`.
- Support des rèsultats multiples.
- Rendre possible la spècification de `long_query_time` avec une granularitè en microsecondes.
- Lier le code de `myisampack` dans le serveur.
- Porter le code de MySQL sur QNX.
- Porter le code de MySQL sur BeOS.
- Porter le code du client MySQL sur LynxOS.
- Ajouter un buffer de clès temporaires durant les commandes `INSERT/DELETE/UPDATE` de maniére a le restaurer proprement si le fichier d'index se remplit.
- Si vous exècutez une commande `ALTER TABLE` sur une table qui est liée symboliquement sur un autre disque, les tables temporaires devraient être crèées dans ce disque.
- Implèmenter un type `DATE/DATETIME` qui gére les donnèes de fuseaux horaires correctement, afin de simplifier la gestion de ces derniers.
- FreeBSD et MIT-pthreads; est-ce que les threads en veille consomment du temps processeur ?
- Vèrifier si les threads verrouillès consomment du temps processeur.
- Corriger le script de configure pour que l'on puisse compiler toutes les librairies (comme `MyISAM`) sans les threads.
- Ajouter une option pour vider pèriodiquement les pages de clès, avec des clès retardèes si on ne les a pas utilisèes depuis un bon moment.
- Permettre les jointures sur des parties de clès (probléme d'optimisation).
- `INSERT SQL_CONCURRENT` et `mysqld --concurrent-insert` pourraient rèaliser des insertions concurrentes á la fin du fichier, si le fichier est verrouillè en lecture.
- Les curseurs cotè serveur.
- Vèrifier si `lockd` fonctionne avec les noyaux moderne. Si ce n'est pas le cas, nous devons corriger `lockd` ! Pour tester cela, dèmarrez `mysqld` avec l'option `--enable-locking` et lancez diffèrents tests de fork*. Si cela fonctionne, il ne doit y avoir aucune erreur.
- Permettre des variables SQL dans la clause `LIMIT`, comme `LIMIT @a,@b`.
- Permettre les modifications de variables dans les commandes `UPDATE`. Par exemple : `UPDATE TABLE foo SET @a=a+b,a=@a, b=@a+c`.

- Changer les modifications de variables utilisateur pour que l'on puisse les utiliser avec la clause `GROUP BY`, comme ceci : `SELECT id, @a:=COUNT(*), SUM(sum_col)/@a FROM nom_de_table GROUP BY id`.
- Ne pas utiliser de valeur par dèfaut `DEFAULT` automatiquement. Gènèrer une erreur lorsque la commande `INSERT` ne contient pas la valeur d'une colonne qui n'a pas de valeur par `DEFAULT`.
- Corriger '`libmysql.c`' pour permettre les commandes `mysql_query()` dans une ligne sans lire les rèsultats, ou afficher une erreur lorsque l'on tente de le faire.
- Vèrifier pourquoi MIT-pthreads `ctime()` ne fonctionne pas sur certains systémes FreeBSD.
- Ajouter une option `IMAGE` á la commande `LOAD DATA INFILE` pour ne pas modifier les colonnes de type `TIMESTAMP` et `AUTO_INCREMENT`.
- Ajouter la syntaxe `LOAD DATE INFILE ... UPDATE`.
  - Pour les tables avec clès primaires, si les donnèes ne contiennent pas de clè primaire, les entrèes qui correspondent á cette clè primaire sont modifièes avec le reste des colonnes. Cependant, les colonnes qui **manquent** dans les donnèes entrantes sont ignorèes.
  - Pour les tables avec des clès primaires á qui il manque une partie de la clè dans les donnèes entrantes, ou si il n'y a pas la clè primaire, les donnèes sont traitèes comme pour `LOAD DATA INFILE ... REPLACE INTO`.
- Faire que la commande `LOAD DATA INFILE` comprenne ceci :

```
LOAD DATA INFILE 'file_name.txt' INTO TABLE tbl_name
     TEXT_FIELDS (text_field1, text_field2, text_field3)
     SET table_field1=CONCAT(text_field1, text_field2),
         table_field3=23
     IGNORE text_field3
```

  Ceci peut être utilisè pour sauter les colonnes de texte supplèmentaire dans le fichier, ou pour modifier des colonnes en se basant sur les donnèes lues.
- `LOAD DATA INFILE 'file_name' INTO TABLE 'nom_de_table' ERRORS TO err_nom_de_table`. Cette commande enregistrera les erreurs et les alertes dans la table `err_nom_de_table`. Cette table aura la structure suivante :

```
line_number    - numèro de ligne dans le fichier de donnèes
error_message  - Le message d'erreur ou d'alerte
Ou peut être
data_line      - La ligne du fichier de donnèes
```

- Affichage automatique de `mysql` vers Netscape.
- `LOCK DATABASES` (avec diffèrentes options).
- Fonctions : ADD_TO_SET(value,set) et REMOVE_FROM_SET(value,set).
- Ajouter les syntaxes `t1 JOIN t2 ON ...` et `t1 JOIN t2 USING ...` Actuellement, vous ne pouvez utiliser cette syntaxe qu'avec `LEFT JOIN`.
- Bien plus de valeurs dans la commande `show status`. Les lignes lues et modifièes. Les sèlections sur une table et les jointures. Le nombre moyen de table dans une sèlection. Le nombre de clause `ORDER BY` et `GROUP BY` dans les requêtes.

- Si vous interrompez `mysql` au beau milieu d'une requête, vous devriez ouvrir une autre connexion, et tuer l'ancienne requête. Alternativement, une tentative de dètection doit avoir lieu sur le serveur.
- Ajouter un gestionnaire d'interface pour les informations de tables, de maniére á pouvoir l'utiliser comme une table systéme. Cela serait un peu lent si vous demandez toutes les informations sur toutes les tables, mais c'est trés souple. `SHOW INFO FROM tbl_name` pour les informations simples de tables devrait être implèmentè.
- `NATURAL JOIN`.
- Permettre la commande `SELECT a FROM crash_me LEFT JOIN crash_me2 USING (a)`; dans ce cas, `a` est supposè provenir de la table `crash_me`.
- Corriger le code pour que `ON` et `USING` fonctionne avec les jointures de type `JOIN`.
- Commande Oracle `CONNECT BY PRIOR ...` pour rechercher les hièrarchies.
- `mysqladmin copy database new-database`; requiert la commande `COPY` dans `mysqld`.
- La liste des processus devrait afficher le nombre de requêtes et de threads.
- `SHOW HOSTS` doit afficher les informations sur les noms d'hôte en cache.
- Options de `DELETE` et `REPLACE` dans la commande `UPDATE` (cela effacerait des lignes lorsque une clè voit un doublon surgir durant la mise á jour).
- Modifier le format de `DATETIME` pour stocker les fractions de secondes.
- Ajout de tous les types de donnèes qui manquent de ANSI92 et ODBC 3.0.
- Changer les noms des tables de chaîne vide en `NULL` pour les colonnes calculèes.
- Ne pas utiliser `Item_copy_string` sur les valeurs numèriques pour èviter la conversion nombre -> chaîne -> nombre dans le cas de : `SELECT COUNT(*)*(id+0) FROM nom_de_table GROUP BY id`
- Rendre possible l'utilisation de la nouvelle librairie d'expression règuliére GNU en lieu et place de celle qui a actuellement cours (la librairie GNU devrait être bien plus rapide).
- Modifier la commande `ALTER TABLE` pour qu'elle n'interrompe pas les clients qui exècutent des commandes `INSERT DELAYED`.
- Corriger le code pour que lorsque les colonnes sont rèfèrencèes dans une commande `UPDATE`, elle contiennent les anciennes valeurs, et non pas les nouvelles.
- Ajouter la simulation de `pread()`/`pwrite()` sous Windows pour permettre les insertions concurrentes.
- Un analyseur de fichiers de log, qui pourrait traiter les donnèes et indiquer quelles tables sont les plus souvent utilisèes, combien de jointures sont faites, etc. Cela aidera les utilisateurs á identifier les points sensibles dans leur conception de bases de donnèes.
- Ajouter `SUM(DISTINCT)`.
- Ajouter les fonctions de groupement `ANY()`, `EVERY()` et `SOME()`. En ANSI SQL, elles ne fonctionnent que sur les colonnes de type boolèen, mais nous pouvons les ètendre pour qu'elles fonctionnent sur n'importe quelle type de colonne ou d'expression en appliquant la conversion valeur == 0 -> FALSE et valeur <> 0 -> TRUE.
- Faire que le type de `MAX(column)` soit le même que le type de la colonne utilisè. Par exemple :

```
mysql> CREATE TABLE t1 (a DATE);
mysql> INSERT INTO t1 VALUES (NOW());
mysql> CREATE TABLE t2 SELECT MAX(a) FROM t1;
mysql> SHOW COLUMNS FROM t2;
```

- Inventer une syntaxe pratique pour une commande qui va modifier avec `UPDATE` une ligne si elle existe, et sinon, faire un insertion `INSERT` si cette ligne n'existe pas. (Comme `REPLACE` le fait avec `INSERT` / `DELETE`).

1.8.4 Ce qui est prèvu pour plus tard

- Implèmenter la fonction : `get_changed_tables(timeout,table1,table2,...)`.
- Remplacer la lecture dans les tables par une zone mèmoire aussi souvent que possible. Actuellement, seules les tables compressèes utilisent des memmap.
- Rendre le code des timestamp automatiques bien plus pratique. Ajouter les timestamps dans le log avec `SET TIMESTAMP=#;`.
- Utiliser un mutex de lecture/ècriture pour gagner de la vitesse.
- Ajouter le support complet des clès ètrangéres pour les tables `MyISAM`, probablement aprés l'implèmentation des procèdures stockèes et des triggers.
- Des vues simples (sur une table, puis sur une expression).
- Fermer automatiquement des tables si une table, une table temporaire ou un fichier temporaire reçoit une erreur 23 (plus assez de fichiers ouverts).
- Lors de la rencontre de field=#, changez toutes les occurrences du champ en #. Actuellement, cela n'est fait que pour les cas simples.
- Changer toutes les expressions constantes par des expressions calculèes, si possible.
- Optimiser la relation clè = expression. Actuellement, seul les relations clè = champ et clè = constante sont optimisèes.
- Fusionner les fonctions de copie pour amèliorer le code.
- Changez '`sql_yacc.yy`' pour le remplacer par un analyseur de ligne de commande plus petit, et qui gére mieux les messages.
- Changer l'analyseur pour utiliser uniquement une régle pour tous les nombres d'arguments possibles dans une fonction.
- Utiliser les calculs de noms complets dans la clause ORDER (pour ACCESS97).
- `MINUS`, `INTERSECT` et `FULL OUTER JOIN`. (actuellement, `UNION` [en 4.0] et `LEFT OUTER JOIN` fonctionnent).
- `SQL_OPTION MAX_SELECT_TIME=#` pour donner une limite de temps á une requête.
- Diriger le log d'historique vers une base.
- Amèliorer `LIMIT` pour permettre la lecture de donnèes á la fin du rèsultat.
- Alertes lors des connexion/ècriture/lecture du client.
- Notez ces modifications de `safe_mysqld` : selon la FSSTND (que Debian essaie de suivre) les fichiers PID devraient être placès dans '`/var/run/<progname>.pid`' et les fichiers de logs dans '`/var/log`'. Il serait bien si vous pouviez mettre le "DATADIR" dans la premiére dèclaration de "pidfile" et "log", de façon á ce que l'emplacement de ces fichiers puisse être modifiè en une seule ligne.

- Permettre au client de commander le log des actions.
- Ajouter l'utilisation de `zlib()` pour les fichiers `gzip`, avec la commande `LOAD DATA INFILE`.
- Corriger le tri et le groupage avec les colonnes `BLOB` (en partie rèsolu).
- Procèdures stockèes. Les triggers sont aussi au programme.
- Un langage de modification simple et atomique, qui peut être utilisè pour ècrire des boucles dans le serveur MySQL.
- Utiliser des sèmaphores pour compter les threads. Il faut commencer par implèmenter des sèmaphores pour MIT-pthreads.
- Ne pas assigner de nouvelle valeur `AUTO_INCREMENT` lorsque l'on utilise la valeur 0. Utilisez `NULL` á la place.
- Ajouter le support complet pour les `JOIN` avec parenthéses.
- Comme alternative á la relation un thread, une connexion, gèrer un groupe de threads pour rèpondre aux requêtes.
- Permettre la pose de plusieurs verrous avec `GET_LOCK`. Lors de ces verrous multiples, gèrer le cas des blocages par verrous qui pourrait être introduit.

Le temps est indiquè en temps de travail et non pas en temps normal.

1.8.5 Ce qui n'est pas prèvu

- Rien. Nous nous dirigeons vers la compatibilitè compléte avec ANSI 92/ANSI 99.

1.9 Comparatif de MySQL avec les autres serveurs SQL

Nos utilisateurs ont rèussi á exècuter nos tests avec un grand nombre de serveurs SQL `Open Source` ou non. Nous connaissons des tests exècutès avec `Oracle`, `DB/2`, `Microsoft SQL Server` et d'autres produits commerciaux. Pour des raisons lègales, nous ne pouvons pas les publier dans notre manuel de rèfèrence.

Cette section inclut la comparaison de MySQL avec `mSQL` pour des raisons historiques, et avec `PostgreSQL` car c'est aussi une base `Open Source`. Si vous avez des rèsultats de tests que nous pouvons publier, contactez-nous á `benchmarks@mysql.com`.

Pour une liste comparative des fonctions et types supportès, voyez la page web de l'utilitaire `crash-me` á l'adresse `http://www.mysql.com/information/crash-me.php`.

1.9.1 MySQL face á `mSQL`

Performance

Pour un vèritable test de vitesse, consultez la suite de test de MySQL. Voir Section 5.1.4 [MySQL Benchmarks], page 360.

Comme il n'y a pas de crèation de threads, que l'analyseur est plus petit, et que les fonctionnalitès sont moins nombreuses, `mSQL` devrait être plus rapide avec :

- les tests qui rèalisent des connexions rèpètèes et qui n'exècutent qu'une simple requête á chaque fois.

- les opèrations d'`INSERT` dans des tables trés simples, avec peu de colonnes et d'index.
- les commandes de `CREATE TABLE` et `DROP TABLE`.
- les commandes `SELECT` sur tout ce qui n'est pas indexè (un scan de table est trés facile).

Comme ces opèrations sont trés simples, il est difficile d'être meilleur lorsque les coûts d'administration sont bien plus forts. Une fois la connexion ètablie, le serveur MySQL devrait être bien plus rapide.

D'un autre cotè, le serveur MySQL est plus rapide que `mSQL` (et que les autres serveurs SQL) avec :

- les opèrations `SELECT` complexes.
- les lectures de grands rèsultats (le serveur MySQL a un protocole plus rapide, plus sûr et bien meilleur).
- les tables á taille de chaînes variable, car le serveur MySQL a une gestion plus efficace et utilise les colonnes `VARCHAR` indexèes.
- la gestion des tables avec de nombreuses colonnes.
- la gestion des tables ayant des grandes lignes.
- les commandes `SELECT` avec de nombreuses expressions.
- les commandes `SELECT` sur de grandes tables.
- la gestion de nombreuses connexions simultanèes. Le serveur MySQL est complétement multi-threadè. Chaque connexion posséde son propre thread, ce qui signifie qu'il y a pas d'attente entre les threads (á moins que l'un d'entre eux modifie une table que l'autre veut utiliser). En `mSQL`, une fois que la connexion a ètè ètablie, les autres doivent attendre que la premiére ait fini, indèpendamment de la durèe d'exècution de la requête. Lorsque la premiére connexion se termine, le suivant peut être servi, tandis que les autres attendent.
- les jointures. `mSQL` peut devenir irrèmèdiablement lent si vous modifiez l'ordre des tables dans la colonne `SELECT`. Dans la suite de test, il s'est vu des tests 15000 fois plus lent que MySQL. Ceci est dû au manque d'optimisateurs chez `mSQL`. Toutefois, si vous avez placè les tables dans le bon ordre avec `mSQL2` et que la clause `WHERE` est simple et utilise les index, la jointure peut être plutôt rapide. Voir Section 5.1.4 [MySQL Benchmarks], page 360.
- les clauses `ORDER BY` et `GROUP BY`.
- la clause `DISTINCT`.
- l'utilisation des colonnes de type `TEXT` et `BLOB`.

Support de SQL

- `GROUP BY` et `HAVING`. `mSQL` ne supporte pas du tout la clause `GROUP BY`. MySQL supporte totalement la clause `GROUP BY` avec les clauses `HAVING` et les fonctions suivantes : `COUNT()`, `AVG()`, `MIN()`, `MAX()`, `SUM()` et `STD()`. `COUNT(*)` est optimisèe pour retourner trés vite le rèsultat si `SELECT` lit les lignes dans une table, qu'aucune autre colonne n'est lue, et qu'il n'y a pas

de clause `WHERE`. `MIN()` et `MAX()` accepte les chaînes de caractéres comme arguments.

- Les `INSERT` et `UPDATE` avec calculs d'expression. MySQL peut faire des calculs d'expression dans les commandes `INSERT` et `UPDATE`. Par exemple :

```
mysql> UPDATE SET x=x*10+y WHERE x<20;
```

- Aliasing. MySQL supporte les alias de colonne.
- Identification simple des colonnes. Avec MySQL, si un nom de colonne est unique dans les tables utilisèes pour une jointure, vous n'avez pas á utiliser son identifiant total.
- `SELECT` avec des fonctions MySQL dispose de nombreuses fonctions (trop nombreuses á lister ici : voir Section 6.3 [Functions], page 436).

Efficacitè d'utilisation de l'espace disque

C'est á dire, comment pouvez-vous rèduire la taille de vos tables ?

MySQL a des types trés prècis, et vous pouvez crèer des tables avec un trés petit espace. Un exemple de type de donnèes pratique est le type `MEDIUMINT` qui occupe 3 octets. Si vous avez 100 millions d'enregistrement, èconomiser un octet par enregistrement est trés important.

`mSQL2` a un jeu de type de donnèes plus limitè, ses tables sont donc plus grosses.

Stabilitè Il est difficile de juger de cela objectivement. Pour un dèbut sur la stabilitè du serveur MySQL, voyez Section 1.2.3 [Stability], page 8.

Nous n'avons aucune expèrience avec la stabilitè du code de `mSQL`, alors nous ne nous prononceront pas lá-dessus.

Prix Un autre point important est la licence. MySQL dispose d'une licence plus souple que `mSQL`, est il est aussi moins cher que `mSQL`. Quelque soit le produit que vous utilisez, n'oubliez pas de prendre un contrat de support par email ou une licence commerciale.

interface Perl

MySQL a la même interface avec Perl que `mSQL`, avec des fonctionnalitès supplèmentaires.

JDBC (Java)

MySQL supporte de nombreux pilotes JDBC :

- le pilote mm : un pilote JDBC de type 4, ècrit par Mark Matthews `mmatthew@ecn.purdue.edu`. Il est publiè sous licence LGPL.
- Le pilote Resin : un pilote JDBC commercial et Open Source. `http://www.caucho.com/projects/jdbc-mysql/index.xtp`
- Le pilote gwe : une interface Java par GWE technologies (non supportè).
- Le pilote jms : un pilote gwe amèliorè par Xiaokun Kelvin ZHU `X.Zhu@brad.ac.uk` (non supportè).
- Le pilote twz : un pilote JDBC de type 4, ècrit par Terrence W. Zellers `zellert@voicenet.com`. Il est commercialisè, mais gratuit pour les utilisations personnelles et èducationnelles (non supportè).

Le pilote recommandè est le pilote mm. Le pilote Resin est aussi bon (les tests sont bons) mais nous n'avons pas reçu beaucoup d'informations á son propos.

Nous savons que `mSQL` a un pilote JDBC, mais nous n'avons pas l'expèrience pour le comparer.

Vitesse de dèveloppement

MySQL a une petite èquipe de dèveloppeur, mais nous sommes trés habituès á coder en C et C++, trés rapidement. Comme les threads, les fonctions, `GROUP BY` et de nombreuses autres fonctionnalitès ne sont pas encore codèes pour `mSQL`, il y a beaucoup faire pour nous rattraper. Pour mesurer cela, vous pouvez consulter le fichier d'historique de `mSQL` '`HISTORY`' pour l'an dernier, et le comparer avec la section news du manuel de rèfèrence MySQL (voir Annexe D [News], page 727). Il est alors assez èvident de voir qui se dèveloppe plus vite.

Programmes utilitaires

Les deux projets `mSQL` et MySQL ont des contributiosns de tiers trés intèressantes. Comme il est plus facile de porter vers les versions plus rècentes (depuis `mSQL` vers MySQL), presque toutes les applications intèressantes de `mSQL` sont disponibles pour MySQL.

MySQL est livrè avec un programme simple `msql2mysql` qui corrige les diffèrences de syntaxe entre `mSQL` et MySQL pour la plupart des fonctions de l'API C. Par exemple, il modifie les occurrences de `msqlConnect()` en `mysql_connect()`. Convertir un programme client de `mSQL` vers MySQL requiert un effort minimal.

1.9.1.1 Comment convertir des outils `mSQL` pour MySQL

Selon notre expèrience, il ne prend pas longtemps pour convertir des outils tels que `msql-tcl` et `msqljava` qui utilisent l'API C de `mSQL` pour qu'ils fonctionnent avec l'API C de MySQL.

La procèdure de conversion est la suivante :

1. Exècutez le script shell `msql2mysql` sur la source. Ce script requiert le programme `replace`, qui est distribuè avec le serveur MySQL.
2. Compilez.
3. Corrigez les erreurs de compilation.

Les diffèrences entre l'API C de `mSQL` et de MySQL sont :

- MySQL utilise la structure `MYSQL` comme type de connexion, alors que `mSQL` utilise un `int`.
- `mysql_connect()` prend un pointeur sur la structure `MYSQL` comme paramétre. Il est facile d'en dèfinir un globalement, ou d'utiliser la fonction `malloc()` pour en obtenir un. `mysql_connect()` prend aussi deux paramétres supplèmentaires, qui sont le nom de l'utilisateur et son mot de passe. Vous pouvez aussi utiliser `NULL, NULL` pour retrouver l'utilisation classique.
- `mysql_error()` prend la structure `MYSQL` comme paramétre. Ajoutez simplement ce paramétre á votre vieux code `msql_error()`, si vous portez du vieux code.

- MySQL retourne un numèro d'erreur et un message d'erreur pour tous les messages d'erreurs. `mSQL` ne retourne que le texte d'erreur.
- Certaines incompatibilitès existent du fait que le serveur MySQL supporte des connexions multiples, dans le même processus.

1.9.1.2 Diffèrences entre les protocoles de communication de `mSQL` et de MySQL

Il y a tellement de diffèrences qu'il est impossible (ou tout au moins trés difficile) de ne pas les supporter toutes.

Les traits les plus caractèristiques de diffèrences entre le protocole de communication de MySQL et de `mSQL` :

- Un buffer de message peut contenir plusieurs lignes.
- Le buffer de message est dynamiquement agrandi si la requête ou le rèsultat sont plus grands que le buffer courant, jusqu'á une taille limite configurable.
- Tous les paquets sont numèrotès pour identifier les paquets doublons ou manquants.
- Toutes les valeurs de colonnes sont èchangèes en mode ASCII. La taille de la colonne est envoyèe dans un paquet binaire (1,2 ou 3 octets).
- MySQL peut lire un rèsultat non bufferisè (sans avoir a stocker tout le rèsultat dans le client).
- Si une opèration de lecture/ècriture prendre plus de 30 secondes, le serveur ferme la connexion.
- Si la connexion est inactive pendant 8 heures, le serveur ferme la connexion.

1.9.1.3 Comparatif des syntaxes SQL de `mSQL` 2.0 et MySQL

Types de colonnes

`Serveur MySQL`
: Dispose des types traditionnels (entre autres : voir Section 6.5.3 [`CREATE TABLE`], page 504):

  - le type `ENUM` pour les chaînes á valeurs limitèes.
  - le type `SET` pour les combinaisons de valeurs.
  - `BIGINT` pour les entiers 64-bit.

  MySQL supporte aussi les attributs de types suivants :

  - option `UNSIGNED` pour les entiers et nombres á virgule flottante.
  - otion `ZEROFILL` pour les entiers.
  - option `AUTO_INCREMENT` pour les colonnes qui sont `PRIMARY KEY`. Voir Section 8.4.3.30 [`mysql_insert_id()`], page 635.
  - valeur par `DEFAULT` pour toutes les colonnes.

`mSQL2`
: Les types de colonnes de `mSQL` correspondent aux types de MySQL dans cette table :

mSQL type	Type MySQL correspondant
CHAR(len)	CHAR(len)
TEXT(len)	TEXT(len). len est la taille maximale. Et LIKE fonctionne.
INT	INT. Avec de nombreuses options supplèmentaires.
REAL	REAL. Ou FLOAT. Les versions 4 et 8 octets sont disponibles.
UINT	INT UNSIGNED
DATE	DATE. Utilise le format ANSI SQL plutôt que le format spècifique de mSQL.
TIME	TIME
MONEY	DECIMAL(12,2). Un nombre á virgule fixe.

Crèation d'index

Serveur MySQL
: Les index peuvent être spècifiès au moment de la crèation de la table avec la commande CREATE TABLE.

mSQL
: Les index doivent être crèès aprés la crèation de la table, avec une commande CREATE INDEX sèparèe.

Insertion d'un identifiant unique dans une table

Serveur MySQL
: Utilisez l'option de colonne AUTO_INCREMENT. Voir Section 8.4.3.30 [mysql_insert_id()], page 635.

mSQL
: Crèez une SEQUENCE dans la table et sèlectionnez la colonne _seq.

Obtenir un identifiant unique pour une ligne

Serveur MySQL
: Ajoutez la clè PRIMARY KEY ou UNIQUE á la table et utilisez ceci. Nouveau en version 3.23.11 : si la clè PRIMARY ou UNIQUE est constituèe d'une seule colonne et qu'il est de type entier, vous pouvez y faire rèfèrence avec _rowid.

mSQL
: Utilisez la colonne _rowid. Notez bien que _rowid peut changer durant ce temps, en fonction de nombreux facteurs.

Lire la date de derniére modification d'une colonne

Serveur MySQL
: Ajoutez une colonne de type TIMESTAMP dans la table. Cette colonne prend automatiquement la date et l'heure courante lors des commandes INSERT ou UPDATE si vous ne lui affectez pas de valeur, ou si vous lui donnez la valeur NULL.

mSQL
: Utilisez la colonne _timestamp.

Comparaisons avec la valeur NULL

Serveur MySQL
: MySQL suit la norme ANSI SQL, et la comparaison avec une valeur NULL retourne toujours NULL.

`mSQL` Avec `mSQL`, `NULL = NULL` est vrai (TRUE). Vous devez changer `=NULL` en `IS NULL` et `<>NULL` en `IS NOT NULL` lorsque vous portez du vieux code de `mSQL` vers MySQL.

Comparaisons de chaînes

`Serveur MySQL`
Normalement, la comparaison de chaînes est faîte sans tenir compte de la casse, et l'ordre de tri est dèterminè par le jeu de caractéres courant (ISO-8859-1 Latin1 par dèfaut). Si vous ne le souhaitez pas, dèclarez vos colonnes avec l'option `BINARY`, qui fera que les comparaisons peuvent être faites en fonction de l'ordre ASCII utilisè sur l'hôte MySQL.

`mSQL` Toutes les comparaisons de chaînes sont faites en tenant compte de la casse, en fonction de l'ordre ASCII.

Recherche insensible á la casse

`Serveur MySQL`
`LIKE` est un opèrateur sensible ou non á la casse, suivant les colonnes utilisèes. Si possible, MySQL utilise des index si l'argument de `LIKE` ne commence pas par un caractére joker.

`mSQL` Utilise `CLIKE`.

Gestion des espaces finaux

`Serveur MySQL`
Supprime tous les espaces de fin de chaînes dans les colonnes `CHAR` et `VARCHAR`. Utilisez les colonnes de type `TEXT` si ce comportement n'est pas souhaitè.

`mSQL` Conserve les espaces finaux.

clause `WHERE`

`Serveur MySQL`
MySQL ètablit correctement les prioritès : `AND` est calculè avant les `OR`. Pour obtenir le comportement de `mSQL` avec MySQL, utilisez les parenthéses (tel que prèsentè dans l'exemple un peu plus loin).

`mSQL èvalue les opèrations de gauche á droite. Cela signifie que`
certaines opèrations logiques exprimèes avec trois arguemnts ou plus ne peuvent être exprimèes. Cela signifie aussi que vous devez modifier des requêtes lorsque vous passez á MySQL : tout ce que vous avez á faire est d'ajouter des parenthéses. Supposez que vous avez le code `mSQL` suivant :

```
mysql> SELECT * FROM table WHERE a=1 AND b=2 OR a=3 AND b=4;
```

Pour le passer sur MySQL et le faire fonctionner comme sur `mSQL`, vous devez ajouter ces parenthéses :

```
mysql> SELECT * FROM table WHERE (a=1 AND (b=2 OR (a=3 AND (b=4))));
```

Contrôle d'accés

`Serveur MySQL`
Posséde des tables qui stockent les droits de chaque utilisateur, hôte et base. Voir Section 4.2.6 [Privileges], page 209.

`mSQL` Dispose d'un fichier '`mSQL.acl`' dans lequel vous pouvez ajouter et modifier les droits d'accés des utilisateurs.

1.9.2 Comparatif de MySQL avec `PostgreSQL`

Lorsque vous lirez ces sections, notez bien que les deux produits èvoluent continuellement. Les èquipes de MySQL AB et de PostGreSQL travaillent toutes deux á rendre leur base les meilleures possibles, et elles prèsentent toutes les deux des alternatives sèrieuses aux bases de donnèes commerciales.

La comparaison qui suit a ètè fait par MySQL AB. Nous avons essayè d'être aussi prècis et objectifs que possible, mais même si nous connaissons MySQL sur le bout des doigts, nous n'avons pas une connaissance aussi pointues des possibilitèes de PostgreSQL : nous pourrions être en erreur. Nous corrigerons ces erreurs dés qu'elles seront portèes á notre attention.

Nous souhaitons noter que PostgreSQL et MySQL sont deux produits largement rèpandus, mais qui prèsentent des objectifs diffèrents, même si tous les deux se dirigent vers la compatibilitè ANSI SQL. Cela signifie que pour certaines applications, MySQL est bien plus optimisè, et pour d'autres, c'est PostGreSQL. Lorsque vous choisissez votre serveur de base de donnèes, commencez par vèrifier que le jeu de fonctionnalitès de votre base correspond bien á votre application. Si vous avez besoin de vitesse pure, MySQL est probablement votre choix. Si vous avez besoin des fonctionnalitès que PostGreSQL propose, utilisez `PostgreSQL`.

1.9.2.1 Stratègies de dèveloppement de MySQL et PostgreSQL

Lorsque nous ajoutons des fonctionnalitès á MySQL, nous mettons un point d'honneur á le faire de maniére optimale et dèfinitive. Le code doit être tellement bon que vous n'y verrez rien á ajouter dans un futur accessible. Nous sommes peu enclins á sacrifier la vitesse aux fonctionnalitès, mais nous faisons notre possible pour trouver une solution qui donnera le meilleur. Cela signifie que le dèveloppement prend plus de temps, mais que le rèsultat en vaut la peine. Ce type de dèveloppement n'est possible que parce que le code est vèrifiè par quelques (actuellement 2) personnes avant qu'il ne soit inclus dans le serveur.

Chez MySQL AB, nous croyons aux publications frèquentes car cela nous permet de transmettre nos nouveautès plus rapidement aux utilisateurs. Comme nous publions une nouvelle version mineure toutes les trois semaines, et une version majeure chaque annèe. Toutes les versions sont testèes soigneusement, avec nos outils de tests et sur de nombreuses plateformes.

PostgreSQL est basè sur un noyau de nombreux contributeurs. Avec cette configuration, il est logique d'ajouter de nombreuses fonctionnalitès, au lieu de les implanter optimalement, car il sera toujours temps d'optimiser lorsque le besoin s'en fera sentir.

Une autre diffèrence majeure entre MySQL et PostgreSQL est que presque tout le code de MySQL est fait par des dèveloppeurs employès par MySQL AB et qui travaillent toujours sur ce code. Les seules exceptions sont le moteur de transaction et la librairie d'expressions règuliéres.

Le contraste est saisissant avec le code de PostgreSQL, dont la majoritè est rèalisèe par un grand groupe de personnes avec diffèrentes sensibilitès. Ce n'est que rècemment que les dèveloppeurs de PostGreSQL ont annoncè qu'ils ont eu finalement le temps de faire le tour du code de la version courante.

Les deux stratègies de dèveloppement prèsentèes ont leur mèrites et inconvènients. Chez MySQL AB, nous pensons, bien sur, que notre modéle est le meilleur car il conduit á une meilleure cohèrence du code, et, á notre avis, moins de bogues. Comme nous sommes les auteurs du serveur MySQL, nous sommes les mieux placès pour coordonner les publications.

1.9.2.2 Comparaison des fonctionnalitès de MySQL et PostgreSQL

Sur la page `crash-me` (`http://www.mysql.com/information/crash-me.php`) vous pouvez trouver une liste des avantages et limitations que l'on peut dètecter avec un programme. Notez bien que de nombreuses èvaluations numèriques peuvent être modifièes avec les options de dèmarrage des diffèrentes bases de donnèes. Cette page web est extrêmement pratique si vous voulez vous assurer que votre application fonctionnera sur diffèrents serveurs SQL, ou si vous voulez convertir votre application d'un serveur á l'autre.

MySQL offre les avantages suivants sur PostgreSQL:

- `MySQL` est gènèralement plus rapide que PostgreSQL. MySQL 4.0.1 a aussi un cache de requêtes qui peut accèlèrer les requêtes pour les lectures de tables les plus frèquentes.
- MySQL a une base d'utilisateurs bien plus grande que PostgreSQL. Par consèquent, le code est testè dans de meilleures conditions, et il a ètè historiquement plus stable que PostgreSQL. MySQL est utilisè plus souvent en environnement de production que PostgreSQL, notamment grâce au fait que MySQL AB, ex TCX DataKonsult AB, a fourni un support commercial de haute qualitè pour le serveur MySQL depuis la premiére publication, tandis que, jusqu'á rècemment, PostgreSQL n'avait pas de support.
- MySQL fonctionne bien mieux sur Windows que PostgreSQL. MySQL fonctionne comme une application Windows native (un service sur NT/2000/XP), tandis que PostgreSQL fonctionne en mode èmulation de `Cygwin`. Nous avons entendu dire que PostgreSQL n'est pas encore stable sous Windows mais non n'avons pas ètè capable de le vèrifier nous-mêmes.
- MySQL a plus d'API avec d'autres langages, et est supportè par un plus grand nombre de programmes que PostgreSQL. Voir Annexe B [Contrib], page 711.
- MySQL fonctionne avec les systémes critiques, á forte charge, 24h/24 et 7j/7. Dans la plupart des cas, nous n'avons jamais á effectuer d'entretien de MySQL. PostgreSQL ne supporte par encore les systémes 24h/24 et 7j/7, car il faut exècuter `VACUUM` une fois de temps en temps pour rècupèrer l'espace vide issu des commandes `UPDATE` et `DELETE`, et pour faire des analyses statistiques qui sont primordiales pour obtenir de bonnes performances avec PostgreSQL. `VACUUM` est aussi nècessaire aprés avoir ajoutè un grand nombre de lignes dans une table. Sur un systéme trés occupè avec de nombreuses modifications, `VACUUM` doit être exècutè trés frèquemment, et parfois, plusieurs fois dans la journèe. Durant l'exècution de `VACUUM`, qui peut prendre plusieurs heures si la base de donnèes est importante, le serveur est, d'un point de vue production, quasiment mort. Notez qu'avec PostgreSQL version 7.2, l'entretien simple des tables ne nècessite plus le verrouillage des tables, et le niveau d'utilisation est alors á peu prés normal. Une

nouvelle commande `VACUUM FULL` effectue les entretiens traditionnels en verrouillant la table, et en rèduisant sa taille sur le disque.

- Le systéme de rèplication de MySQL a ètè testè soigneusement, et il est utilisè sur des sites comme :
  - Yahoo Finance (`http://finance.yahoo.com/`)
  - Mobile.de (`http://www.mobile.de/`)
  - Slashdot (`http://www.slashdot.org/`)
- Dans les deux distributions de MySQL, vous trouverez des suites de tests, '`mysql-test-run`' et `crash-me` (`http://www.mysql.com/information/crash-me.php`), ainsi que la suite de performances. Le systéme de tests est souvent modifiè avec des nouveaux tests pour les nouvelles fonctionnalitès, ainsi que les bogues qui nous sont rapportès. Nous testons chacune de nos versions sur un grand nombre de plate-formes. Ces tests sont plus sophistiquès que tout ce que nous avons pu voir chez PostgreSQL, et il nous assurent que le serveur MySQL reste de bonne qualitè.
- Il y a bien plus de livres concernant MySQL que PostgreSQL. O'Reilly, SAMS, Que et New Riders sont des èditeurs importants, avec des livres traitant de MySQL. Toutes les fonctionnalitès de MySQL sont aussi documentèes sur le manuel en ligne, car lorsqu'une nouvelle fonctionnalitè est implèmentèe, les dèveloppeurs MySQL doivent la documenter avant qu'elle soit inclue dans le source.
- MySQL support plus de fonctions ODBC que `PostgreSQL`.
- MySQL a une commande `ALTER TABLE` beaucoup plus puissante.
- MySQL supporte les tables sans transaction pour les applications qui ont besoin de plus de vitesse. Les tables peuvent être placèes en mèmoire, comme les tables `HEAP`, ou bien sur le disque, comme les tables `MyISAM`. Voir Chapitre 7 [Table types], page 531.
- MySQL supporte deux types de gestionnaires de tables, qui supportent les transactions : `InnoDB` et `BerkeleyDB`. Comme chaque moteur de transactions travaille dans diffèrentes conditions, cela donne plus de possibilitès aux auteurs d'applications pour choisir leur solution optimale pour la configuration du serveur, au besoin table par table. Voir Chapitre 7 [Table types], page 531.
- Les tables groupèes `MERGE` vous donnent un moyen d'accèder instantanèment a plusieurs tables de même structure, et de les utiliser ensemble, comme une seule. C'est le moyen parfait lorsque vous avez des tables de logs trop grosses, et que vous les sèparez par mois. Voir Section 7.2 [MERGE], page 539.
- La possibilitè de compresser les tables en lecture seule, mais tout en ayant un accés direct aux lignes de la table, vous donnent plus de performance en lecture, en rèduisant la taille des donnèes lues sur le disque. C'est trés pratique pour les archives. Voir Section 4.7.4 [`myisampack`], page 297.
- MySQL posséde le support d'outils de recherche en texte plein. Voir Section 6.8 [Full-text Search], page 522.
- Vous pouvez accèder á de nombreuses bases de donnèes avec la même connexion, du moment que vous avez les droits pour le faire.
- MySQL est codè depuis le dèbut pour être multi-threadè, tandis que PostgreSQL utilise les processus. Le temps de changement de contexte et d'accés aux zones de stockage est bien plus court avec des threads qu'entre des processus sèparès. Cela donne un

avantage de vitesse á MySQL dans les applications multi-utilisateurs, et aussi avec les systémes á plusieurs processeurs symètriques.

- MySQL a un systéme de droits bien plus avancè que PostgreSQL. Alors que PostgreSQL ne supporte que des droits pour les commandes `INSERT`, `SELECT` et `UPDATE/DELETE` au niveau de l'utilisateur pour une base ou une table, MySQL vous permet de dèfinir un jeu de droits complet pour une base, une table ou une colonne. MySQL vous permet aussi de spècifier des droits par hôte ou utilisateur. Voir Section 4.3.1 [GRANT], page 226.
- MySQL supporte un protocole client/serveur compressè qui amèliore grandement les performances sur les connexions lentes.
- MySQL applique le concept des "gestionnaires de table", et est la seule base de donnèes relationnelle que nous connaissons qui utilise ce concept. Cela permet diffèrents types de tables depuis le moteur SQL, et chaque table peut être optimisèe pour diffèrentes performances.
- Tous les types de tables MySQL (exceptè `InnoDB`) sont implèmentès avec des fichiers (une table par fichier), ce qui rend trés souple les sauvegardes, les dèplacements et les effacements, ou même l'utilisation des liens symboliques pour les tables et les bases, même lorsque le serveur est èteint.
- Les outils pour rèparer et optimiser les tables `MyISAM` (le type de tables MySQL le plus courant). Un outil de rèparation n'est nècessaire que pour les corruptions physiques dans un fichier de donnèes, et gènèralement, pour corriger une dèfaillance du matèriel. Elle permet á la majoritè des donnèes d'être rècupèrèes.
- Changer de version de MySQL est trés simple. Lorsque vous changer de version de MySQL, vous n'avez pas á exporter et rèimporter vos donnèes, comme vous devez le faire avec PostgreSQL.

Les inconvènients de MySQL face á PostgreSQL:

- Le support des transactions de MySQL n'est pas aussi bien testè que celui de PostgreSQL.
- Comme le serveur MySQL utilise les threads, qui ne sont pas encore parfaitement rodès sur tous les systémes d'exploitation, il faut alors utiliser les exècutables binaires de `http://www.mysql.com/downloads/`, ou bien lire avec beaucoup de soin les instructions dans la section Section 2.3 [Installing source], page 84 pour obtenir un exècutable qui fonctionne de maniére optimale.
- Le verrouillage de table, tel qu'utilisè par les tables non transactionnelles `MyISAM`, est dans de nombreux cas plus rapide que les verrous de page, de ligne ou le versionnage. L'inconvènient est que si l'on ne prend pas garde au fonctionnement du verrouillage de table, une requête seule, á longue durèe d'exècution peut bloquer la table en ècriture pour un long moment. Cela peut être èvitè en concevant correctement l'application. Sinon, on peut toujours changer le gestionnaire de la table pour un modéle transactionnel. Voir Section 5.3.2 [Table locking], page 381.
- Avec UDF (user-defined functions, fonction dèfinies par les utilisateurs), il est possible de faire èvoluer les fonctions du serveur MySQL aussi bien pour les fonctions SQL normales que pour les fonctions d'agrègation, mais ce systéme n'est pas aussi pratique que pour PostgreSQL. Voir Section 9.2 [Adding functions], page 672.
- Les modifications qui s'effectuent sur plusieurs tables sont bien plus difficiles á faire avec MySQL. Toutefois, cela a ètè corrigè avec la version 4.0.2 et les commandes `UPDATE`

multi-tables, et en MySQL version 4.1, avec les sous-sèlections. En MySQL version 4.0, on peut utiliser les effacements multi-tables pour effacer plusieurs lignes dans diffèrentes tables simultanèment. Voir Section 6.4.6 [DELETE], page 494.

PostgreSQL offre actuellement les avantages suivants sur MySQL :

Notez que comme nous connaissons le plan de dèveloppement de MySQL, nous avons inclus la table suivante, avec les version du serveur MySQL oú les fonctionnalitès seront supportèes. Malheureusement, nous ne pouvions pas faire les mêmes comparaisons avec les fonctionnalitès á venir de PostgreSQL, car nous ne connaissons pas le plan de dèveloppement de PostgreSQL.

Fonctionnalitè	**version de MySQL**
Sous-sèlections	4.1
Clè ètrangéres	5.0 (3.23 avec InnoDB)
Vues	5.0
Procèdures stockèes	5.0
Triggers	5.0
Unions	4.0
Jointures totales	4.1
Contraintes	4.1 ou 5.0
Curseurs	4.1 ou 5.0
R-trees	4.1 (pour les tables MyISAM)
Tables hèritèes	Non prèvu
Type de donnèes extensible	Non prèvu

Autres raisons de considèrer PostgreSQL :

- L'utilisation standard de PostgreSQL est parfois plus proche de la norme ANSI SQL.
- Il est possible d'accèlèrer les traitements de PostgreSQL en codant des procèdures stockèes.
- Pour les donnèes gèographiques, les R-trees font de PostgreSQL un meilleur choix que MySQL. (note : MySQL version 4.1 dispose des R-trees pour les tables MyISAM).
- L'optimiseur PostgreSQL fait des optimisations que l'optimiseur de MySQL ne peut faire. La plus importante est lors de jointures, lorsqu'il n'y a pas les clès nècessaires, ou lorsque les jointures se font avec des clès combinèes par un OR. La suite de tests de MySQL `http://www.mysql.com/information/benchmarks.html` montre quels types de constructions vous devriez èviter lorsque vous utilisez diffèrentes bases de donnèes.
- PostgreSQL a une plus grande èquipe de dèveloppement qui contribue au serveur.

Les inconvènients de PostgreSQL face á MySQL :

- L'utilitaire `VACUUM` rend PostgreSQL difficile á utiliser en environnement de production 24h/24 et 7j/7.
- Les tables sont obligatoirement transactionnelles.
- Les commandes `INSERT`, `DELETE` et `UPDATE` sont plus lentes.

Pour une liste compléte des inconvènients, examinez la premiére table de cette section.

1.9.2.3 Performances comparèes de MySQL et PostgreSQL

Le seul test de vitesse `Open Source` que nous connaisssons et qui peut être utilisè pour tester les performances de MySQL et de PostgreSQL (et d'autres bases de donnèes) est le nôtre. Il est disponible á l'adresse `http://www.mysql.com/information/benchmarks.html`.

Nous avons demandè de nombreuses fois aux dèveloppeurs de PostgreSQL et aux utilisateurs de PostgreSQL de nous aider á dèvelopper et amèliorer ce test, pour en faire le meilleur test possible, mais nous n'avons pas reçu de rèponse de leur part.

Nous, les dèveloppeurs MySQL, avons, á cause de cela, passè de nombreuses heures á obtenir les meilleures performances de PostgreSQL pour ces tests, mais comme nous ne connaissons pas intimement le serveur PostgreSQL, nous sommes certains d'être passè á cotè de certains points. Sur la page de test, nous avons documentè exactement comment nous avons exècutè le test, pour qu'il soit facile á tout le monde de reproduire et de confirmer nos rèsultats.

Les tests sont gènèralement exècutès avec et sans l'option `--fast`. Lorsque l'option `--fast` est utilisèe, nous tentons d'utiliser tous les trucs connus du serveur pour que l'exècution soit aussi rapide que possible. L'idèe est que le mode normal doit montrer comment le serveur devrait se comporter avec une configuration normale, et le mode `--fast` montre comment le serveur se comporte si le dèveloppeur utilise toutes les extensions du serveur pour accèlèrer son application.

Lorsque nous exècutons PostgreSQL avec l'option `--fast`, nous exècutons la commande `VACUUM` aprés chaque modification importante de la table, et aprés l'effacement de table, pour conserver la base dans un ètat parfait pour les prochaines sèlections. Le temps nècessaire á l'exècution de la commande `VACUUM` est mesurè sèparèment.

Lorsque PostgreSQL 7.1.1 fonctionne, nous n'avons pu le faire fonctionner avec l'option `--fast` car durant les test d'`INSERT`, le postmaster (dèmon PostgreSQL) s'est arrètè, et la base ètait tellement corrompue qu'il a ètè impossible de redèmarrer le postmaster. Aprés un deuxiéme probléme identique, nous avons dècidè de reporter les tests `--fast` jusqu'á la prochaine version de PostgreSQL. Les dètais sur la machine avec laquelle nous avons exècutè les tests sont disponible sur la page de tests.

Avant de passer aux autres tests de performances que nous connaissons, nous souhaitons vous donner des dètails sur le contexte des tests.

Il est trés facile d'ècrire un test qui montre que **toutes** les bases du monde sont les meilleures, en se limitant simplement aux points forts de la base, et en ne testant rien qui soit une faiblesse. De plus, si la conclusion se raméne á un seul chiffre, c'est encore plus facile.

Cela revient á mesurer la vitesse de MySQL par rapport á PostgreSQL en regardant simplement les temps moyens des tests de MySQL sur notre page. En se basant sur ces informations, notre serveur MySQL serait 40 fois plus rapide que PostgreSQL, ce qui est, bien sûr, faux. Nous pourrions même rendre les choses encore pires en prenant les tests oú PostgreSQL a les pires performances, et dire que MySQL est plus de 2000 fois plus rapide que PostgreSQL.

Le fait est que MySQL fait un certain nombre d'optimisation que PostgreSQL ne fait pas. C'est aussi vrai, bien sûr, dans le sens inverse. Un optimiseur SQL est un programme complexe, et une compagnie peut passer des annèes á le rendre de plus en plus rapide.

Lorsque vous regardez les rèsultats des tests, vous devriez vous limiter aux opèrations que vous utilisez dans votre application, et utiliser simplement ces rèsultats pour dècider quelle

base de donnèes est la plus adaptèe á vos besoins. Les rèsultats de tests vous montrent quelles opèrations sont le point faible de chaque base, et vous donnent des indications sur ce qu'il faudra èviter de faire.

Nous connaissons deux tests de performances qui indiquent que PostgreSQL est plus rapide que MySQL. Ces deux tests sont faits dans un environnement multi-utilisateur, un test que nous, chez MySQL AB, nous n'avons pas eu le temps de mettre en place car c'est une tâche ènorme de le faire pour que ce soit èquitable pour chaque base.

Le premier est le test payant, effectuè pour Great Bridge, une entreprise qui a tentè durant 16 mois de bâtir un modéle d'affaires avec PostgreSQL mais qui a dèsormais cessè les opèrations. C'est probablement le pire test que nous ayons jamais vu. Non seulement ce test ètait bâti pour ne tester que les points forts de PostgreSQL, mais il ètait totalement dèloyal pour toutes les autres bases de donnèes utilisèes dans le test.

Note : nous savons aussi que même certains des dèveloppeurs principaux de PostgreSQL n'ont pas aimè la maniére avec laquelle Great Bridge a dirigè le test, et nous ne blâmons pas l'èquipe de PostgreSQL pour ces tests.

Ce test a ètè condamnè dans de nombreux articles et groupes d'utilisateurs, et nous ne citerons que quelques-uns des points qui ètaient incorrects.

- Les tests ont ètè exècutès avec un outil commercial trés cher, qui rendent impossibles aux entreprises `Open Source` comme nous de vèrifier les rèsultats ou même de voir comment les tests ont ètè mis en place. L'outil n'ètait même pas un vèritable outil de test de performances, mais une application de mise en place et de tests. Dire que c'est un outil "standard" de tests est bien loin de la vèritè.
- Great Bridge a admis qu'ils ont optimisè le serveur PostgreSQL (avec `VACUUM` avant le test) et optimisè le dèmarrage pour les tests, ce qu'ils n'ont pas fait pour les autres bases. Ils disent "Ce processus optimise les index et libére un peu d'espace sur le disque. Les index optimisès amèliorent les performances marginalement". Nos tests indiquent clairement la diffèrence lorsqu'on exècute une sèrie de sèlections sur une base qui a ètè entretenue avec `VACUUM` ou pas : le facteur peut atteindre 10.
- Les rèsultats des tests ètaient eux-même trés ètranges. La documentation de tests AS3AP mentionne qu'ils sont des "sèlections, jointures simples, projections, agrègations, modification d'une ligne et modifications massives".

  PostgreSQL est bon pour les `SELECT`s et les `JOIN`s (spècialement aprés l'exècution de la commande `VACUUM`), mais n'est pas trés performant pour les `INSERT`s ou les `UPDATE`. Les tests semblent indiquer que seuls des `SELECT` ont ètè exècutès. Cela peut facilement expliquer le nombre de bons rèsultats pour PostgreSQL dans ce test. Les mauvais rèsultats de MySQL seront èvident un peu plus bas, dans ce document.
- Ils ont exècutè les soi-disant tests depuis une machine sous Windows vers un serveur Linux, via ODBC, une configuration que personne n'utiliserait pour un environnement multi-utilisateur. Cela testait surtout le pilote ODBC et le protocole Windows utilisè entre les clients et le serveur, plutôt que le serveur lui-même.
- Lorsque les tests ont ètè exècutès avec Oracle et MS-SQL (Great Bridge a indirectement indiquè que ces bases ètaient utilisèes dans les tests), ils n'ont pas utilisè le protocole natif mais ODBC. Toute personne qui a utilisè Oracle sait que toutes les vraies applications utilisent l'interface native au lieu d'ODBC. Faire un test via ODBC et dire que les rèsultats sont comparables avec les situations d'utilisation de la base dans le monde

rèel ne peut être considèrè comme èquitable. Ils auraient dû faire deux tests, avec et sans ODBC, pour fournir des faits (aprés que leurs experts aient optimisè les serveurs, bien sûr).

- Ils font rèfèrence au tests TPC-C, mais ils ne mentionnent nulle part que le test n'ètait pas un vrai test TPC-C et qu'ils n'ètaient pas autorisès á dire que c'ètait un tel test. Un test TPC-C ne peut être conduit que par les régles approuvèes par le TPC Council (`http://www.tpc.org/`). Great Bridge ne l'a pas fait. Ils ont donc violè la marque commerciale de TPC et discrèditè leurs propres tests. Les régles dictèes par le TPC Council sont trés strictes pour s'assurer que personne ne produit de faux rèsultats ou tire des conclusions invèrifiables. Apparemment, Great Bridge n'ètait pas interessè par cela.
- Aprés le premier test, nous avons contactè Great Bridge et nous leur avons mentionnè les erreurs èvidentes qu'ils avaient faits avec le serveur MySQL :
  - Travailler avec une version beta de notre pilote ODBC.
  - Travailler avec un systéme Linux qui n'ètait pas optimisè pour les threads.
  - Utiliser une vieille version de MySQL alors qu'il en existait une nouvelle.
  - Ne pas dèmarrer MySQL avec les bonnes options pour gèrer les environnements fortement multi-utilisateurs (la configuration par dèfaut est optimisèe pour consommer le moins de ressources).

  Great Bridge a exècutè un nouveau test, avec notre nouveau pilote ODBC, et avec de meilleures options de dèmarrage de MySQL mais ils ont refusè d'utiliser notre librairie glibc ou notre binaire standard (utilisè par 80% de nos utilisateurs), qui ètaient statiquement compilè avec la librairie glibc.

  Selon nos sources, Great Bridge n'a rien fait pour s'assurer que les autres bases de donnèes ètaient correctement configurèes pour leur environnement de tests. Nous sommes certains qu'ils n'ont pas contactè Oracle ou Microsoft pour leur demander des conseils de configurations.
- Le test ètait payè par Great Bridge, et ils ont dècidè de publier une partie des rèsultats, bien choisis (au lieu de tous les publier).

Tim Perdue, un inconditionnel de PostgreSQL et un utilisateur rèticent de MySQL, publia une comparaison sur PHPbuilder (`http://www.phpbuilder.com/columns/tim20001112.php3`).

Lorsque nous avons eu connaissance de la comparaison, nous avons tèlèphonè á Tim Perdue á ce propos, car il y avait de nombreux rèsultats ètranges dans ses publications. Par exemple, il statuait que MySQL avait un probléme avec 5 utilisateurs durant ses tests, alors que nous avons des utilisateurs avec des machines similaires, qui utilisent MySQL avec plus de 2000 utilisateurs simultanès, et 400 requêtes par secondes. (Dans ce cas, la limite est la bande passante du rèseau, pas le serveur SQL).

Il semblait qu'il utilisait un noyau Linux qui avait des problémes avec les threads, comme les noyaux avant le 2.4 qui avait des problémes avec les threads sur les machines á plusieurs processeurs. Nous avons documentè dans le manuel comment corriger ce probléme, et Tim devrait être bientôt au courant de ce probléme.

Les autres problémes possibles pourraient provenir d'une vieille librairie glibc ou du fait que MySQL n'utilise pas une version exècutable de notre site web, qui est bâtie avec une version corrigèe de glibc library. Dans ces cas, les symptômes sont ceux que Tim a mesurè.

Nous avons demandè á Tim d'avoir accés a ses donnèes pour que nous puissions reproduire le test, et s'il pouvait vèrifier la version de MySQL sur sa machine pour voir ce qui n'allait pas. Il a promis qu'il nous contacterait mais n'a rien fait á ce jour.

A cause de cela, nous ne pouvons pas non plus avoir confiance en ce test.

Avec le temps qui passe, les technologies èvoluent, et les tests prècèdents ne sont dèsormais plus valables. Le serveur MySQL dispose d'un jeu de gestionnaires de tables, avec des choix trés diffèrents sur la vitesse et les fonctionnalitès. Voir Chapitre 7 [Table types], page 531. Il serait intèressant de voir comment les tests ci-dessus s'effectuent avec les diffèrents gestionnaires de tables de MySQL. PostgreSQL a aussi, bien sûr, de nouvelles fonctionnalitès depuis que ces tests ont ètè menès. Comme ces tests ne sont pas publiquement disponibles, il n'y a pas moyen pour nous de savoir comment les bases se comportent de nos jours avec ces tests.

Conclusion :

Les seuls tests qui existent aujourd'hui, disponibles au tèlèchargement pour tous, et que vous pouvez exècuter avec MySQL, PostgreSQL sont les tests de MySQL. Chez MySQL AB, nous croyons que les bases de donnèes `Open Source` devraient être testèes avec des outils `Open Source` ! C'est le seul moyen de nous assurer que personne ne peut faire de tests que personne ne peut reproduire, et les utiliser pour proclamer que sa base est meilleure que les autres. Sans connaître le fond des choses, il est impossible de rèpondre á une telle dèclaration.

Ce que nous trouvons ètrange est que chaque test que nous avons vu á propos de PostgreSQL est impossible á reproduire, et indique que PostgreSQL est le meilleur dans la plupart des tests, alors que les nôtres , que tout le monde peut reproduire, montrent le contraire. Avec cela, nous ne voulons pas dire que PostgreSQL n'est pas bon dans de nombreuses tâches (il l'est !!), ou qu'il n'est pas plus rapide que MySQL dans certaines conditions. Nous voulons juste voir un test èquitable oú PostgreSQL se comporte bien, pour que nous puissions avoir une compètition amicale !

Pour plus d'informations sur nos tests, voyez Section 5.1.4 [MySQL Benchmarks], page 360.

Nous travaillons actuellement sur une suite de tests de perfomance qui inclue des tests multi utilisateurs, et une meilleure documentation sur les tests individuels pour indiquer ce qu'ils font et comment ajouter d'autres tests á la suite.

2 Installation de MySQL

Ce chapitre dècrit comment obtenir et installer MySQL :

- Pour une liste de sites á partir desquels vous pouvez obtenir MySQL, voyez Section 2.2.2 [Getting MySQL], page 73.
- Pour savoir quelles plate-formes sont supportèes, voyez Section 2.2.3 [Which OS], page 74. Veuillez noter que tous les systémes supportès ne sont pas tous aussi bons pour faire fonctionner MySQL. Certains sont plus robustes et efficaces que d'autres, voyez Section 2.2.3 [Which OS], page 74 pour davantage d'informations.
- Plusieurs versions de MySQL sont disponibles á la fois sous forme de binaires ou de code source. Nous fournissons aussi un accés public á notre arbre source aux personnes souhaitant suivre les derniers dèveloppements et nous aider á tester le nouveau code. Pour dèterminer quelle version et quel type de distribution vous devriez utiliser, voyez Section 2.2.4 [Which version], page 76. En cas de doute, utilisez la distribution binaire.
- Les instructions d'installation á partir des binaires ou des sources sont donnèes dans Section 2.2.8 [Installing binary], page 82, et Section 2.3 [Installing source], page 84. Chacun de ces jeux d'instructions inclut une partie sur des problémes spècifiques que vous pourriez rencontrer.
- Pour les procèdures consècutives á l'installation, voyez Section 2.4 [Post-installation], page 97. Ces procèdures s'appliquent á la fois si vous installez MySQL á partir des sources ou á partir des binaires.

2.1 Installation standard rapide de MySQL

2.1.1 Installer MySQL sous Linux

Il est recommandè d'installer MySQL sous Linux en utilisant un fichier RPM. Les RPM de MySQL sont actuellement compilè sur une RedHAt en version 6.2, mais devraient fonctionner sur toute autre version de Linux qui supporte `rpm` et utilise `glibc`.

Si vous avez des problémes avec un fichier RPM, si vous obtenez par exemple l'erreur "`Sorry, the host 'xxxx' could not be looked up`", rèfèrez vous á Section 2.6.1.1 [Binary notes-Linux], page 117.

Les fichiers RPM dont vous pourriez avoir besoin sont :

- `MySQL-VERSION.i386.rpm`

  Le serveur MySQL. Vous en aurez besoin á moins que vous ne vouliez que vous connecter á un serveur MySQL tournant sur une autre machine.

- `MySQL-client-VERSION.i386.rpm`

  Les programmes clients MySQL standards. Vous avez certainement besoin d'installer ce paquet.

- `MySQL-bench-VERSION.i386.rpm`

  Tests et bancs d'essai. Nècessite Perl et les modules RPM msql-mysql-modules.

- `MySQL-devel-VERSION.i386.rpm`

  Librairies et fichiers d'inclusions dont vous aurez besoin pour compiler d'autres clients MySQL, tels que les modules Perl.

- `MySQL-VERSION.src.rpm`

  Celui-ci contient le code source de tous les paquets prècèdents. Il peut donc être utilisè pour construire des fichiers RPM pour d'autres architectures (par exemple, l'Alpha ou le SPARC).

Pour voir tous les fichiers prèsents dans un paquet RPM, lancez :

```
shell> rpm -qpl MySQL-VERSION.i386.rpm
```

Pour effectuer une installation standard minimale, lancez :

```
shell> rpm -i MySQL-VERSION.i386.rpm MySQL-client-VERSION.i386.rpm
```

Pour installer uniquement le paquet du client MySQL, lancez :

```
shell> rpm -i MySQL-client-VERSION.i386.rpm
```

Le fichier RPM place les donnèes dans '`/var/lib/mysql`'. Le RPM crèe aussi les entrèes appropriès dans '`/etc/rc.d/`' pour lancer le serveur automatiquement au dèmarrage. (Cela signifie que, si vous avez dèjá effectuè une installation auparavant, vous pouvez avoir besoin de faire une sauvegarde de vos fichiers de dèmarrage prècèdents si vous les changez, de façon á ne pas les perdre.)

Aprés l'installation des fichiers RPM, le dèmon `mysqld` devrait fonctionner et vous devriez être capables de commencer á utiliser MySQL. Voir Section 2.4 [Post-installation], page 97.

Si quelque chose se passe mal, vous pouvez trouver davantage d'informations dans le chapitre d'installation des binaires. Voir Section 2.2.8 [Installing binary], page 82.

2.1.2 Installer MySQL sous Windows

Le serveur MySQL pour Windows est disponible sous deux formes :

1. La forme binaire contient un programme de configuration qui installe tout ce dont vous avez besoin de telle façon que vous puissiez dèmarrer le serveur immèdiatement.
2. La distribution sous forme de code source contient tout le code et les fichiers de support nècessaires á la compilation des exècutables en utilisant le compilateur VC++ 6.0. Voir Section 2.3.7 [Windows source build], page 96.

En régle gènèrale, vous devriez utiliser la distribution binaire.

Vous aurez besoin des choses suivantes :

- Un systéme d'exploitation Windows 32-bit tel que Windows 9x, Me, NT, 2000, ou XP. La famille NT (NT, Windows 2000 et XP) permet de faire fonctionner le serveur MySQL en tant que service. Voir Section 2.6.2.2 [NT start], page 122.

  Si vous voulez utiliser des tables de plus de 4 Go, vous devez installer MySQL sur un systéme de fichiers NTFS ou supèrieur. N'oubliez pas d'utiliser `MAX_ROWS` et `AVG_ROW_LENGTH` quand vous crèez les tables. Voir Section 6.5.3 [CREATE TABLE], page 504.

- Support du protocole TCP/IP.

- Une copie du binaire ou de la distribution MySQL pour WIndows, qui peut être tèlèchargèe sur `http://www.mysql.com/downloads/`.

  Note : les fichiers de la distribution sont dans un fichier au format zip et nous vous recommandons l'utilisation d'un client FTP adèquat qui supporte la reprise des transferts afin d'èviter des problémes de corruption de fichiers pendant le tèlèchargement.
- Un programme d'extraction `ZIP` pour dècompresser le fichier de la distribution.
- Suffisamment d'espace disque pour dècompresser, installer et crèer les bases de donnèes en relation avec vos besoins.
- Si vous envisagez de vous connecter á MySQL via `ODBC`, vous aurez aussi besoin du pilote `MyODBC`. Voir Section 8.3 [ODBC], page 599.

2.1.2.1 Installation des binaires

1. Si vous travaillez avec les serveurs NT/2000/XP, loguez-vous en utilisateur avec les permissions d'administrateur.
2. Si vous faites une mise á jour d'une version plus rècente de MySQL, il est nècessaire d'arrêter le serveur. Si vous utilisez le serveur en tant que service, utilisez :

   ```
   C:\> NET STOP MySQL
   ```

   Dans les autres cas, utilisez :

   ```
   C:\mysql\bin> mysqladmin -u root shutdown
   ```
3. Avec les serveurs NT/2000/XP, si vous voulez changer l'exècutable du serveur (e.g., -max or -nt), il est aussi nècessaire de stopper le service :

   ```
   C:\mysql\bin> mysqld-max-nt --remove
   ```
4. Dèzipper les fichiers dans un emplacement temporaire.
5. Exècutez le fichier '`setup.exe`' pour commencer l'installation. Si vous voulez l'installer á un autre emplacement que celui pas dèfaut '`c:\mysql`', utilisez le bouton `Parcourir` pour spècifier un emplacement.
6. Terminer le processus d'installation.

2.1.2.2 Prèparation de l'environnement MySQL de Windows

A partir de la version 3.23.38 de MySQL, la distribution Windows inclut le binaire du serveur normal et le binaire du serveur MySQL-Max. Voici une liste des diffèrents serveurs MySQL que vous pouvez utiliser :

Binaire	Description
`mysqld`	Compilè avec dèbogage total et vèrification automatique de l'allocation de mèmoire, liens symboliques, tables InnoDB et BDB.
`mysqld-opt`	Binaire optimisè sans le support des tables transactionnelles.
`mysqld-nt`	Binaire optimisè pour NT/2000/XP sans support des tunnels nommès. Vous pouvez faire fonctionner cette version sur Windows 9x/Me, mais dans ce cas, les tunnels nommès ne sont pas crèes et vous devez avoir TCP/IP installè.
`mysqld-max`	Binaire optimisè avec support des liens symboliques et des tables InnoDB et BDB.

`mysqld-max-nt` Comme `mysqld-max`, mais compilè avec le support des tunnels nommès.

A partir de la version 3.23.50, les tunnels nommès ne sont activès que si on dèmarre mysqld avec `--enable-named-pipe`.

Tous les binaires prècèdents sont optimisès pour le processeur Pentium Pro mais devraient fonctionner avec n'importe quel processeur Intel >= i386.

Vous devrez utiliser un fichiers d'options pour spècifier votre configuration MySQL dans les circonstances suivantes :

- Le rèpertoire d'installation ou de donnèes est diffèrent de ceux par dèfaut ('`c:\mysql`' et '`c:\mysql\data`').
- Vous voulez utilisez l'un de ces serveurs :
  - mysqld.exe
  - mysqld-max.exe
  - mysqld-max-nt.exe
- Vous avez besoin de paramètrer la configuration du serveur.

Normalement, vous pouvez utiliser l'outil `WinMySQLAdmin` pour èditer le fichier d'options `my.ini`. Dans ce cas, vous n'avez pas á vous soucier de ce qui suit.

Il y'a deux fichiers d'options avec la même fonction : '`my.cnf`' et '`my.ini`'. Toutefois, pour èviter la confusion, il est prèfèrable de n'en utiliser qu'un seul. Les deux fichiers sont en texte plein. Le fichier '`my.cnf`', si utilisè, doit être crèe dans le rèpertoire racine du lecteur C. Le fichier '`my.ini`', si utilisè, doit être crèe dans le rèpertoire systéme de Windows. (Ce rèpertoire est souvent nommè '`C:\WINDOWS`' ou '`C:\WINNT`'. Vous pouvez dèterminer sa valeur exacte en regardant la valeur de la variable d'environnement `windir`.) MySQL regarde en premier le fichier `my.ini`, puis le fichier '`my.cnf`'.

Si votre PC utilise un systéme de boot ou le lecteur C n'est pas le lecteur de boot, votre seule option est d'utiliser le fichier '`my.ini`'. Notez aussi que si vous utilisez l'outil `WinMySQLAdmin`, celui-ci n'utilisera que le fichier '`my.ini`'. Le rèpertoire '`\mysql\bin`' contient un fichier d'aide avec les instruction pour utiliser cet outil.

En utilisant `notepad.exe`, crèez le fichier d'options et èditez la section `[mysqld]` pour spècifier les valeurs des paramétres `basedir` et `datadir` :

```
[mysqld]
# set basedir to installation path, e.g., c:/mysql
basedir=le_dossier_d'installation
# set datadir to location of data directory,
# e.g., c:/mysql/data or d:/mydata/data
datadir=le_dossier_des_donnèes
```

Notez que les chemins de Windows doivent être spècifiès dans les fichiers d'options en utilisant des slashes normaux (/) au lieu des backslashes (\). Si vous utilisez des backslashes, vous devez les doubler.

Si vous voulez utiliser un rèpertoires de donnèes autre que '`c:\mysql\data`', vous devez copier le contenu entier du dossier '`c:\mysql\data`' au nouvel endroit.

Si vous voulez utiliser les tables transactionnelles `InnoDB`, vous aurez besoin de crèer manuellement deux nouveaux rèpertoires pour contenir les les donnèes et les logs

InnoDB, '`c:\ibdata`' et '`c:\iblogs`'. Vous aurez aussi besoin d'ajouter quelques lignes supplèmentaires dans le fichier d'options. Voir Section 7.5.2 [InnoDB start], page 545.

Si vous ne voulez pas utiliser les tables `InnoDB`, ajoutez l'option `skip-innodb` au fichier d'options.

Maintenant, vous pouvez commencer á tester le dèmarrage du serveur.

2.1.2.3 Dèmarrer le serveur pour la premiére fois

Tester á partir d'une console DOS est la meilleure chose á faire car le serveur affiche des messages qui y apparaissent. Si quelque chose n'est pas bon dans votre configuration, ces messages vous aiderons á identifier et corriger le probléme.

Assurez-vous d'être dans le rèpertoire oú se situe le serveur, puis entrez cette commande :

```
C:\mysql\bin> mysqld-max --standalone
```

Vous devriez voir ce qui suit pendant le dèmarrage du serveur :

```
InnoDB: The first specified datafile c:\ibdata\ibdata1 did not exist:
InnoDB: a new database to be created!
InnoDB: Setting file c:\ibdata\ibdata1 size to 209715200
InnoDB: Database physically writes the file full: wait...
InnoDB: Log file c:\iblogs\ib_logfile0 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile0 size to 31457280
InnoDB: Log file c:\iblogs\ib_logfile1 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile1 size to 31457280
InnoDB: Log file c:\iblogs\ib_logfile2 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile2 size to 31457280
InnoDB: Doublewrite buffer not found: creating new
InnoDB: Doublewrite buffer created
InnoDB: creating foreign key constraint system tables
InnoDB: foreign key constraint system tables created
011024 10:58:25  InnoDB: Started
```

Pour plus d'informations á propos du dèmarrage de MySQL sous Windows, voyez Section 2.6.2 [Windows], page 121.

2.2 Notes gènèrales á propos de l'installation

2.2.1 Mèthodes d'installation

2.2.2 Comment obtenir MySQL ?

Visitez le site de MySQL (`http://www.mysql.com/`) pour des informations á propos de la version courante et les instructions de tèlèchargement.

Notre miroir principal est situè sur `http://mirrors.sunsite.dk/mysql/`.

Pour une liste compléte et á jour des miroirs web/tèlèchargement de MySQL, voyez `http://www.mysql.com/downloads/mirrors.html`. Vous trouverez lá des informations á propos des futurs miroirs et de quoi nous informer de la non-validitè de l'un d'entre eux.

2.2.3 Systémes d'exploitation supportès par MySQL

Nous utilisons `GNU Autoconf`, alors il est possible de porter MySQL sur tous les systémes modernes qui utilisent les threads Posix et un compilateur C++. Pour compiler uniquement le client, un compilateur C++ est simplement nècessaire. Nous utilisons et dèveloppons le logiciel nous-même, en commençant par Sun Solaris (Versions 2.5 - 2.7) et SuSE Linux version 7.x.

Notez que pour de nombreux systémes d'exploitation, le support natif des threads ne fonctionne qu'avec les derniéres versions. MySQL a ètè compilè avec succés sur les combinaisons systéme d'exploitation/package de threads suivants :

- AIX 4.x, 5.x avec les threads natifs. Voir Section 2.6.6.4 [IBM-AIX], page 139.
- Amiga.
- BSDI 2.x avec le package MIT-pthreads. Voir Section 2.6.4.5 [BSDI], page 135.
- BSDI 3.0, 3.1 et 4.x avec les threads natifs. Voir Section 2.6.4.5 [BSDI], page 135.
- DEC Unix 4.x avec les threads natifs. Voir Section 2.6.6.6 [Alpha-DEC-UNIX], page 141.
- FreeBSD 2.x avec le package MIT-pthreads. Voir Section 2.6.4.1 [FreeBSD], page 133.
- FreeBSD 3.x et 4.x avec les threads natifs. Voir Section 2.6.4.1 [FreeBSD], page 133.
- HP-UX 10.20 avec les threads DCE ou avec le package MIT-pthreads. Voir Section 2.6.6.2 [HP-UX 10.20], page 138.
- HP-UX 11.x avec les threads natifs Voir Section 2.6.6.3 [HP-UX 11.x], page 138.
- Linux 2.0+ avec LinuxThreads 0.7.1+ ou `glibc` 2.0.7+. Voir Section 2.6.1 [Linux], page 113.
- Mac OS X. Voir Section 2.6.5 [Mac OS X], page 136.
- NetBSD 1.3/1.4 Intel et NetBSD 1.3 Alpha (requiert GNU make). Voir Section 2.6.4.2 [NetBSD], page 134.
- OpenBSD > 2.5 avec les threads natifs. OpenBSD < 2.5 avec le package MIT-pthreads. Voir Section 2.6.4.3 [OpenBSD], page 134.
- OS/2 Warp 3, FixPack 29 et OS/2 Warp 4, FixPack 4. Voir Section 2.6.7 [OS/2], page 147.
- SGI Irix 6.x avec les threads natifs. Voir Section 2.6.6.8 [SGI-Irix], page 144.
- Solaris 2.5 et plus rècent, avec les threads natifs sur SPARC et x86. Voir Section 2.6.3 [Solaris], page 129.
- SunOS 4.x avec le package MIT-pthreads. Voir Section 2.6.3 [Solaris], page 129.
- Caldera (SCO) OpenServer avec un port rècent du package FSU Pthreads. Voir Section 2.6.6.9 [Caldera], page 145.
- Caldera (SCO) UnixWare 7.0.1. Voir Section 2.6.6.10 [Caldera Unixware], page 147.
- Tru64 Unix

- Windows 9x, Me, NT, 2000 et XP. Voir Section 2.6.2 [Windows], page 121.

Notez que toutes les plate-formes ne sont pas èquipèes de la même façon pour faire fonctionner MySQL. Les capacitès d'une plateforme pour supporter de fortes charges avec MySQL est dèterminè par ceci :

- Stabilitè gènèrale de la librairie de threads. Une plateforme qui a une excellente rèputation en gènèral, mais une librairie de threads instable, dont le code est utiisè par MySQL, même si le reste est parfait, fera de MySQL une application instable.
- La capacitè du noyeau et/ou de la librairie de threads de profiter des capacitès multiprocesseurs, symètrique ou pas. En d'autres termes, lorsqu'un processus crèe un thread, il doit être possible pour ce thread de s'exècuter sur diffèrents processeurs.
- La capacitè du noyau et/ou de la librairie de threads de faire fonctionner de nombreux threads qui posent et lèvent des verrous mutex en peut de temps, frèquemment, sans changement de contexte excessif. En d'autres termes, si l'implèmentation de `pthread_mutex_lock()` est trop soucieux du temps CPU, cela va ralentir sèrieusement MySQL. Si ce probléme n'est pas règlè, ajouter des processeurs supplèmentaires va finalement ralentir MySQL.
- Performance et stabilitè gènèrale du systéme de fichiers.
- La capacitè du systéme d'exploitation de gèrer de grands fichiers, et de le faire efficacement, si vos tables sont grandes.
- Notre niveau d'expertise avec la plate-forme, chez MySQL AB. Si vous connaissons bien une plate-forme, vous pourront introduire des optimisations et des corrections spècifiques á la plate-forme, et activè lors de la compilation. Nous pouvons aussi fournir des conseils judicieux pour cnofigurer votre systéme optimalement pour MySQL.
- Le temps de tests que vous avons consacrè á des configurations similaires, en interne.
- Le nombre d'utilisateur de MySQL qui font fonctionner MySQL avec succés sur cette plate-forme, avec des configurations similaires. SI ce nombre est grand, les chances de rencontrer un probléme spècifique sont faibles.

En se basant sur les critéres prècèdents, les meilleures plate-forme pour MySQL sont x86 avec SuSE Linux 7.1, 2.4 kernel, et ReiserFS (ou toute autre distribution Linux similaire) et SPARC avec Solaris 2.7 ou 2.8. FreeBSD vient en troisiéme, mais nous espèrons bien le voir rejoindre le groupe de tête, une fois que la librairie de threads sera amèliorèe. Nous espèrons aussi être bientôt capables d'ajouter les autres plate-formes sur laquelle MySQL compile, et fonctionne correctement, mais pas toujours le bon niveau de stabilitè et de performances. Cela rèclame des efforts de notre part, en coopèration avec les dèveloppeurs de ces plateformes. Si vous êtes interessès par l'amèlioration de ces composants, et que vous êtes en position pour influencer le dèveloppement, demandez des instructions dètaillèes á MySQL en envoyant un email á `internals@lists.mysql.com`.

Notez bien que la comparaison prècèdente ne signifie pas qu'un systéme d'exploitation est meilleur que l'autre, en gènèral. Nous avons classè les systémes en fonction de leur capacitè á faire fonctionner un systéme MySQL, et nous nous limitons á cette comparaison. Avec cela en tête, le rèsultat de cette comparaison serait diffèrent si nous y ajoutions d'autres problémes. Et dans certains cas, la seule raison qui fait qu'un OS est meilleur que l'autre est parce que nous y avons consacrè plus de temps, pour optimiser et tester. Nous nous bornons á exprimer notre point de vue pour vous aider á dècider quelle plate forme choisir pour votre serveur MySQL.

2.2.4 Quelle version de MySQL utiliser ?

La premiére dècision á prendre est de savoir si vous voulez utiliser la derniére version de dèveloppement ou la derniére version stable :

- En régle gènèrale, si vous utilisez MySQL pour la premiére fois ou si vous essayer de le porter vers un systéme pour lequel il n'existe pas de distribution binaire, nous vous recommandons d'utiliser la derniére version stable (actuellement la version 3.23). Notez que toutes les versions de MySQL sont passèes aux bancs de test MySQL avant chaque sortie (même les versions de dèveloppement).
- D'autre part, si vous utilisez un vieux systéme et que vous voulez procèder á une mise á jour, sans pour autant risquer de mettre á jour sans raison, vous devriez mettre á jour vers la derniére version de la même branche que celle que vous êtes en train d'utiliser (dans le cas oú un numèro de version supèrieur existe). Nous avons essayè de rèsoudre uniquement les bogues fatals et de produire des patchs petits et sûrs pour cette version.

La seconde dècision est de dèterminer si vous devez utiliser une distribution source ou binaire. Dans la plupart des cas, vous devrez utiliser une distribution binaire, si celle-ci existe pour votre plate-forme, car elle s'installera certainement plus facilement qu'une distribution source.

Dans les cas suivants, vous devrez opter pour une installation á partir des sources :

- Si vous voulez installer MySQL á un endroit particulier. (Les distributions binaires standards sont "prêtes á l'emploi" á n'importe quel endroit, mais vous pouvez vouloir davantage de flexibilitè).
- Pour être apte á satisfaire les diffèrents besoins des utilisateurs, nous fournissons deux distributions binaires : une compilèe avec le support du gestionnaire de tables non transactionnel (un binaire petit et rapide), et une autre configurèe avec la plupart des options avancèes, telles que le support des transactions. Les deux versions sont compilèes á partir des mêmes sources. Tous les clients natifs `MySQL` peuvent se connecter aux deux versions.

  La distribution binaire avancèe est marquèe du suffixe `-max` et est configurèe avec les mêmes options que `mysqld-max`. Voir Section 4.7.5 [`mysqld-max`], page 304.

  Si vous souhaitez utiliser des fichiers RPM de MySQL-Max, vous devez tout d'abord installer les RPM MySQL standards.
- Si vous voulez configurer `mysqld` avec certaines options supplèmentaires qui ne sont pas prèsentes dans les distributions binaires standards. Ci-dessous, une liste des options les plus courantes que vous pourriez vouloir utiliser :
  - `--with-innodb`
  - `--with-berkeley-db`
  - `--with-raid`
  - `--with-libwrap`
  - `--with-named-z-lib (Ceci est appliquè á certains binaires)`
  - `--with-debug[=full]`
- La distribution binaire standard est normalement compilèe avec le support de tous les jeux de caractéres et sont supposès fonctionner sur nombre de processeurs de la même famille.

Si vous souhaitez un serveur MySQL plus vèloce, vous pouvez vouloir le recompiler juste avec le support des jeux de caractéres dont vous avez besoin, utiliser un meilleur compilateur (comme `pgcc`), ou utiliser des options de compilations davantage optimisèes pour votre processeur.

- Si vous avez rencontrè des bogues et que vous les avez rapportè á l'èquipe de dèveloppement MySQL, vous recevrez probablement un patch que vous devrez appliquer á la distribution source pour que le bogue soit rèsolu.
- Si vous voulez consulter (et/ou modifier) le code C et C++ qui compose MySQL, vous devrez obtenir une distribution source. Le code source est toujours la solution ultime. Les distributions sources contiennent aussi davantage de tests et d'exemples que les distributions binaires.

La politique de nommage de MySQL utilise des numèros de version qui consiste en trois nombres suivis d'un suffixe. Par exemple, une version nommèe `mysql-3.21.17-beta` doit être interprètèe de la façon suivante :

- Le premier nombre (`3`) dècrit le format de fichier. Toutes les versions 3 ont le même format de fichier.
- Le second nombre (`21`) correspond au niveau de version. Normalement, il y a le choix entre deux d'entre eux. L'un correspond á la version/branche stable (actuellement `23`) et l'autre se rèfére á la branche de dèveloppement (actuellement `4.0`). Normalement, les deux versions sont stables, mais la version de dèveloppement peut comporter des lacunes, manquer de documentation sur des nouvelles fonctionnalitès, ou peut ne pas compiler sur certains systémes.
- Le troisiéme nombre (`17`) est le numèro de version au sein du niveau de version. Celui-ci est incrèmentè á chaque nouvelle publication. En temps normal, vous souhaiterez utiliser la derniére version du niveau de version que vous avez choisi.
- Le suffixe (`beta`) indique le niveau de stabilitè de la version. Les suffixes possibles sont :
  - `alpha` indique que la publication contient de grandes portions de nouveau code qui n'a pas ètè testè á 100%. Les bogues connus (d'ordinaire, il n'y en a aucun) doivent être documentès dans la section News. Voir Annexe D [News], page 727. Il existe aussi de nouvelles commandes et extensions dans la plupart des versions alpha. Du dèveloppement actif qui inclut des changements majeurs dans le code peut concerner les versions alpha, mais tout sera testè avant de faire une publication. Il ne devrait pas y avoir de bogues connus dans les publications de MySQL.
  - `beta` signifie que tout le nouveau code a ètè testè. Aucune fonctionnalitè majeure qui pourrait causer corruption du code n'est ajoutèe. Il ne doit pas y avoir un seul bogue connu. Une version alpha passe en beta quand il n'y a pas eu de bogue fatal rapportè depuis au moins un mois et que nous ne prèvoyons pas de nouvelle fonctionnalitè qui pourrait corrompre d'anciennes commandes.
  - `gamma` est une version bêta qui existe depuis un certain temps et qui semble fonctionner correctement. Seulement des changements mineurs sont effectuès. C'est ce que de nombreuses autres compagnies appellent une publication.
  - S'il n'y a pas de suffixe, cela signifie que la version fonctionne depuis un certain temps sur diffèrents sites avec aucun rapport de bogue autre que des bogues

spècifiques á une plate-forme. Seuls des corrections critiques sont appliquèes á la publication. C'est ce que l'on appelle une version stable.

Toutes les versions de MySQL passent par nos tests et bancs d'essais standards pour nous assurer qu'elles peuvent être utilisèes sans danger. Les sèries de tests s'amèliorent en permanence car les tests standards sont ètendus dans le temps pour traquer tous les bogues prècèdemment trouvèes.

Notez bien que toutes les versions de MySQL ont ètè testèes au moins avec :

Une batterie de tests internes
: C'est une reproduction d'un systéme de production client. Il comporte de nombreuses tables avec des centaines de Mo de donnèes.

Les bancs de tests MySQL
: Ils effectuent une sèrie de requêtes communes. C'est aussi un test pour savoir si le dernier processus d'optimisation rend le code plus rapide. Voir Section 5.1.4 [MySQL Benchmarks], page 360.

Le test `crash-me`
: Il tente de dèterminer de quelles fonctionnalitès disposent les bases de donnèes et quelles en sont les limites. Voir Section 5.1.4 [MySQL Benchmarks], page 360.

Un autre test provient du fait que nous avons la version la plus rècente de MySQL dans notre propre environnement de production interne, sur au moins une machine. Nous avons plus de 100 Go de donnèes á manipuler.

2.2.5 Dispositions d'installation

Cette section dècrit les rèpertoires par dèfaut crèes en installant les distributions binaires et les distributions de sources.

Une distribution binaire est installèe en la dècompressant dans le rèpertoire d'installation que vous avec choisis (souvent '`/usr/local/mysql`') et crèe les rèpertoires suivants á cet endroit :

Rèpertoire	**Contenu du rèpertoire**
'`bin`'	Programmes clients et le serveur `mysqld`
'`data`'	Fichiers de log et bases de donnèes
'`include`'	Fichiers (entêtes) inclus
'`lib`'	Librairies
'`scripts`'	`mysql_install_db`
'`share/mysql`'	Fichiers de messages d'erreurs
'`sql-bench`'	Benchmarks

Une distribution des sources est installèe aprés que vous l'ayez configurè et compilè. Par dèfaut, l'installation se fait dans '`/usr/local`', dans les sous-rèpertoires suivants :

Rèpertoire	**Contenu du rèpertoire**
'`bin`'	Programmes clients et scripts
'`include/mysql`	Fichiers (entêtes) inclus
'`info`'	Documentation au format Info

‘`lib/mysql`’	Librairies
‘`libexec`’	Le serveur `mysqld`
‘`share/mysql`’	Fichiers de messages d’erreurs
‘`sql-bench`’	Benchmarks et test `crash-me`
‘`var`’	Fichiers de log et bases de données

Dans le rèpertoire d’installation, les dispositions d’une installation des sources différe d’une installation binaire des façons suivantes :

- Le serveur `mysqld` est installè dans le rèpertoire ‘`libexec`’ plutôt que dans le rèpertoire ‘`bin`’.
- Le rèpertoire des donnèes est ‘`var`’ plutôt que ‘`data`’.
- `mysql_install_db` est installè dans le rèpertoire ‘`/usr/local/bin`’ plutôt que dans ‘`/usr/local/mysql/scripts`’.
- Le rèpertoire des fichier d’entête et les rèpertoires des librairies sont ‘`include/mysql`’ et ‘`lib/mysql`’ au lieu de ‘`include`’ et ‘`lib`’.

Vous pouvez crèer votre propre installation binaire á partir d’une distribution de sources compilèes en exècutant le script ‘`scripts/make_binary_distribution`’.

2.2.6 Quand et comment sont publièes les nouvelles versions

MySQL èvolue rapidement ici, á MySQL AB, et nous voulons le partager avec les autres utilisateurs de MySQL. Nous essayons de faire une nouvelle version á chaque fois que nous avons implantè des fonctionnalitès qui seront utiles á d’autres.

Nous essayons aussi d’aider les utilisateurs dont les requêtes sont faciles á programmer. Nous prenons en considèration tout ce que nos clients nous demandent, et nous accordons une attention particuliére á nos clients qui ont pris une licence e-mail ètendue.

Personne n’est obligè de tèlècharger une nouvelle version. La sections News vous indiquera si la nouvelle version contient une fonctionnalitè que vous attendez. Voir Annexe D [News], page 727.

Nous utilisons la politique suivante, lors de la mise á jour de MySQL :

- Pour chaque modification mineure, le dernier numèro de la chaîne de version est incrèmentè. Lorsqu’il y a des nouvelles fonctionnalitès importantes ou des incompatibilitè mineures avec la version prècèdente, nous incrèmentons le chiffre du milieu. Lorsque le format de fichier change, le premier numèro est incrèmentè.
- Des versions tables et testèes sont publièes une á deux fois dans l’annèe, mais si de petits bogues apparaissent, une version qui ne va corriger que ces bogues sera publièe.
- Des versions fonctionnelles avec des corrections de bogues pour les vieilles versions sont publièes toutes les 1 á 8 semaines.
- Les distributions binaires de certaines plate-formes seront compilèes par nos soins pour les versions majeures. D’autres personnes font des versions binaires pour d’autres systémes, mais probablement moins frèquemment.
- Nous rendons gènèralement public les correctifs, une fois que nous avons dècouverts de petits bogues. Ils sont postès sur la liste `bugs@lists.mysql.com` et seront ajoutès á la prochaine version.

- Pour les bogues non critiques mais ennuyant, nous allons les ajouter dans le serveur de sources de MySQL, et ils seront publiès dans la prochaine version.
- Si il y a un bogue fatal dans une version, nous publierons une nouvelle version aussitôt que possible. Nous apprècions que les autres èditeurs fasse la même chose.

La version stable actuelle est la version 3.23; nous avons dèjá commencè á dèvelopper activement la version 4.0. Les bogues seront corrigès dans cette version. Nous ne croyons pas á un gel complet, car cela laisserait en plan des corrections de bogues, et des listes de souhaites. "Un peu gelè" signifie que nous pourrions apporter de petits ajouts qui "n'affecterons pas presque sûrement pas le fonctionnement".

MySQL utilise un systéme de numèrotation lègérement diffèrent pour les autres produits. En gènèral, il est relativement sûr d'utiliser une version qui a ètè publiée il y a quelques semaines sans être remplacèe par une nouvelle version.

2.2.7 Binaires compilès par MySQL AB

Comme service, nous, MySQL AB, proposons un jeu de distributions binaires de MySQL qui sont compilès sur nos machines, ou les machines auxquelles nos clients nous ont gentiment donnè accés.

Ces distributions sont gènèrèes avec le script `scripts/make_binary_distribution` et sont configurès avec les compilateurs et options suivantes :

SunOS 4.1.4 2 sun4c avec `gcc` 2.7.2.1

```
CC=gcc CXX=gcc CXXFLAGS="-O3 -felide-constructors" ./configure
--prefix=/usr/local/mysql --disable-shared --with-extra-
charsets=complex --enable-assembler
```

SunOS 5.5.1 (et plus) sun4u avec `egcs` 1.0.3a ou 2.90.27 ou gcc 2.95.2 et plus rècent

```
CC=gcc CFLAGS="-O3" CXX=gcc CXXFLAGS="-O3 -felide-constructors
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql
--with-low-memory --with-extra-charsets=complex --enable-assembler
```

SunOS 5.6 i86pc avec `gcc` 2.8.1

```
CC=gcc CXX=gcc CXXFLAGS=-O3 ./configure --prefix=/usr/local/mysql
--with-low-memory --with-extra-charsets=complex
```

Solaris 2.8 sparc avec `gcc 2.95.3`

```
CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXX=gcc CXXFLAGS="-O3
-fno-omit-frame-pointer -felide-constructors -fno-exceptions
-fno-rtti" ./configure --prefix=/usr/local/mysql "--with-
comment=Official MySQL binary" --with-extra-charsets=complex
"--with-server-suffix=" --enable-thread-safe-client --enable-
local-infile --enable-assembler --disable-shared
```

Linux 2.0.33 i386 avec `pgcc` 2.90.29 (`egcs` 1.0.3a)

```
CFLAGS="-O3 -mpentium -mstack-align-double" CXX=gcc CXXFLAGS="-O3
-mpentium -mstack-align-double -felide-constructors -fno-
exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql
--enable-assembler --with-mysqld-ldflags=-all-static --with-extra-
charsets=complex
```

Linux 2.2.x avec x686 avec gcc 2.95.2

```
CFLAGS="-O3 -mpentiumpro" CXX=gcc CXXFLAGS="-O3 -mpentiumpro
-felide-constructors -fno-exceptions -fno-rtti" ./configure
--prefix=/usr/local/mysql --enable-assembler --with-mysqld-
ldflags=-all-static --disable-shared --with-extra-charsets=complex
```

SCO 3.2v5.0.4 i386 avec gcc 2.7-95q4

```
CC=gcc CXX=gcc CXXFLAGS=-O3 ./configure --prefix=/usr/local/mysql
--with-extra-charsets=complex
```

AIX 2 4 avec gcc 2.7.2.2

```
CC=gcc CXX=gcc CXXFLAGS=-O3 ./configure --prefix=/usr/local/mysql
--with-extra-charsets=complex
```

OSF/1 V4.0 564 alpha avec gcc 2.8.1

```
CC=gcc CFLAGS=-O CXX=gcc CXXFLAGS=-O3 ./configure --prefix=/usr/local/mysql
--with-low-memory --with-extra-charsets=complex
```

Irix 6.3 IP32 avec gcc 2.8.0

```
CC=gcc CXX=gcc CXXFLAGS=-O3 ./configure --prefix=/usr/local/mysql
--with-extra-charsets=complex
```

BSDI BSD/OS 3.1 i386 avec gcc 2.7.2.1

```
CC=gcc CXX=gcc CXXFLAGS=-O ./configure --prefix=/usr/local/mysql
--with-extra-charsets=complex
```

BSDI BSD/OS 2.1 i386 avec gcc 2.7.2

```
CC=gcc CXX=gcc CXXFLAGS=-O3 ./configure --prefix=/usr/local/mysql
--with-extra-charsets=complex
```

FreeBSD 4.4-stable i386 avec gcc 2.95.3

```
CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXX=gcc CXXFLAGS="-O3
-fno-omit-frame-pointer -felide-constructors -fno-exceptions
-fno-rtti" ./configure --prefix=/usr/local/mysql "--with-
comment=Official MySQL binary" --with-extra-charsets=complex
"--with-server-suffix=" --enable-thread-safe-client --enable-
local-infile --enable-assembler --with-named-z-libs=not-used
--disable-shared
```

Si vous avez des options plus optimales pour l'une des configurations prècèdement listèes, vous pouvez toujours nous en faire part sur la liste de distribution des dèveloppeurs (`internals@lists.mysql.com`).

Les distributions RPM antèrieures á la version 3.22 de MySQL sont contribuèes par nos utilisateurs. A partir de la version 3.22, les RPM sont gènèrès par nous chez MySQL AB.

Si vous voulez compiler une version de dèbogage de MySQL, vous devez ajouter `--with-debug` ou `--with-debug=full` aux lignes de configurations prècèdentes et effacer les options `-fomit-frame-pointer`.

Pour les distributions Windows, voyez Section 2.1.2 [Windows installation], page 70.

2.2.8 Installer MySQL á partir d'une distribution binaire

Vous pouvez aussi vous rèfèrer á Section 2.1.2.1 [Windows binary installation], page 71, Section 2.1.1 [Linux-RPM], page 69, et Section 8.4.7 [Building clients], page 660.

Vous aurez besoin des outils suivants pour installer les binaires MySQL :

- GNU `gunzip` pour dècompresser la distribution.
- Un programme `tar` pour dèsarchiver la distribution. GNU `tar` est connu pour fonctionner. Le `tar` de Sun connaît quelques problémes.

L'utilisation de fichiers RPM (RedHat Package Manager) est une autre façon d'installer MySQL sous Linux. Voir Section 2.1.1 [Linux-RPM], page 69.

Si vous rencontrez des problémes, **utilisez toujours** `mysqlbug` pour poster des questions á `mysql@lists.mysql.com`. Même si le probléme n'est pas un bogue, `mysqlbug` rassemble des informations sur le systéme qui aidera les autres á rèsoudre votre probléme. En n'utilisant pas `mysqlbug`, vous amoindrissez vos chances d'obtenir une solution á votre probléme ! Vous trouverez `mysqlbug` dans le rèpertoire '`scripts`' aprés avoir dèsarchivè la distribution. Voir Section 1.6.2.3 [Bug reports], page 27.

Les commandes de base que vous devez lancer pour installer et utiliser MySQL á partir des binaires sont les suivantes :

```
shell> groupadd mysql
shell> useradd -g mysql mysql
shell> cd /usr/local
shell> gunzip < /path/to/mysql-VERSION-OS.tar.gz | tar xvf -
shell> ln -s full-path-to-mysql-VERSION-OS mysql
shell> cd mysql
shell> scripts/mysql_install_db
shell> chown -R root  .
shell> chown -R mysql data
shell> chgrp -R mysql .
shell> bin/safe_mysqld --user=mysql &
or
shell> bin/mysqld_safe --user=mysql &
si vous utilisez MySQL 4.x
```

Vous pouvez ajouter des nouveaux utilisateurs en utilisant le script `bin/mysql_setpermission` si vous installez les modules Perl `DBI` et `Msql-Mysql-modules`.

Une description plus dètaillèe est disponible ci-dessous.

Pour installer une distribution binaire, suivez ces ètapes et procèdez á Section 2.4 [Post-installation], page 97, pour la configuration et les tests consècutifs á l'installation :

1. Choisissez le rèpertoire ou vous voulez dèsarchiver la distribution et dèplacez-vous y. Dans l'exemple suivant, nous dèsarchivons la distribution dans le rèpertoire '`/usr/local`' et crèons un rèpertoire '`/usr/local/mysql`' dans lequel MySQL est installè. (Les instructions suivantes supposent bien sûr que vous avez les autorisations suffisantes pour crèer des fichiers dans '`/usr/local`'. Si ce rèpertoire est protègè, vous aurez besoin de faire l'installation en tant que `root`.)

2. Procurez-vous un fichier de distribution á partir d'un des sites citès dans Section 2.2.2 [Getting MySQL], page 73.

   Les distributions binaires de MySQL sont fournies sous forme d'archives `tar` compressèes et ont des noms de la forme '`mysql-VERSION-OS.tar.gz`', oú `VERSION` est un nombre (par exemple, `3.21.15`), et `OS` indique le type de systéme d'exploitation pour lequel la distribution est compilèe (par exemple, `pc-linux-gnu-i586`).

3. Si vous voyez une distribution binaire avec le suffixe `-max`, cela signifie que le binaire supporte les tables avec les transactions ainsi que d'autres fonctionnalitès. Voir Section 4.7.5 [`mysqld-max`], page 304. Veuillez noter que tous les binaires sont compilès á partir du même code source.

4. Ajoutez un utilisateur et un groupe avec les droits desquels `mysqld` fonctionnera :

   ```
   shell> groupadd mysql
   shell> useradd -g mysql mysql
   ```

   Ces commandes ajoutent le groupe `mysql` group et l'utilisateur `mysql`. La syntaxe de `useradd` et de `groupadd` peut varier de façon significative suivant les versions d'Unix. Elles peuvent aussi s'appeler `adduser` et `addgroup`. Vous pouvez aussi souhaiter nommer le groupe et l'utilisateur autrement que `mysql`.

5. Dèplacez-vous dans le rèpertoire d'installation choisi :

   ```
   shell> cd /usr/local
   ```

6. Dècompressez la distribution et crèez le rèpertoire d'installation :

   ```
   shell> gunzip < /chemin/de/mysql-VERSION-OS.tar.gz | tar xvf -
   shell> ln -s chemin-complet-de-mysql-VERSION-OS mysql
   ```

   La premiére commande crèe un rèpertoire '`mysql-VERSION-OS`'. La seconde commande crèe un lien symbolique vers ce rèpertoire. Cela vous permet de vous rèfèrer plus facilement au rèpertoire d'installation en appelant '`/usr/local/mysql`'.

7. Dèplacez vous dans le rèpertoire d'installation :

   ```
   shell> cd mysql
   ```

   Vous y trouverez de nombreux fichiers et sous-rèpertoires. Les plus importants lors de l'installation sont les sous-rèpertoires '`bin`' et '`scripts`'.

   '`bin`' Ce rèpertoire contient les programmes clients ainsi que le serveur. Vous devriez ajouter le chemin complet de ce rèpertoire á votre variable d'environnement `PATH` de telle façon que votre interprèteur de commandes trouve les programmes facilement. Voir Annexe F [Environment variables], page 828.

   '`scripts`' Ce rèpertoire contient le script `mysql_install_db` utilisè pour initialiser la base de donnèes `mysql` contenant les tables de priviléges servant á gèrer les permissions d'accés au serveur.

8. Si vous souhaitez utiliser `mysqlaccess` et placer la distribution MySQL á un endroit non standard, vous devez changer l'endroit oú `mysqlaccess` s'attend á trouver le client `mysql`. Editez le script '`bin/mysqlaccess`' aux environs de la ligne 18. Cherchez une ligne qui ressemble á ceci :

   ```
   $MYSQL     = '/usr/local/bin/mysql';    # path to mysql executable
   ```

Changez le chemin pour avoir le bon endroit oú `mysql` est actuellement stockè sur votre systéme. Si vous ne le faites pas, vous obtiendrez une erreur `Broken pipe` quand vous lancerez `mysqlaccess`.

9. Crèez les tables de priviléges MySQL (ceci est nècessaire si vous n'avez jamais installè MySQL auparavant) :

   ```
   shell> scripts/mysql_install_db
   ```

   Veuillez noter que les versions de MySQL antèrieures á la version 3.22.10 dèmarraient le serveur MySQL quand vous exècutiez `mysql_install_db`. Ce n'est plus le cas.

10. Nommez `root` en tant que propriètaire des binaires et l'utilisateur avec les droits duquel vous ferez fonctionner `mysqld` comme propriètaire du rèpertoire de donnèes :

    ```
    shell> chown -R root  /usr/local/mysql
    shell> chown -R mysql /usr/local/mysql/var
    shell> chgrp -R mysql /usr/local/mysql
    ```

    La premiére commande change les attributs `owner` en faveur de l'utilisateur `root`, la seconde change les attributs `owner` du rèpertoire de donnèes en faveur de l'utilisateur `mysql`, et la troisiéme change les attributs `group` en faveur du groupe `mysql`.

11. Si vous voulez installer le support de l'interface Perl `DBI/DBD`, rèfèrez-vous á Section 2.7 [Perl support], page 149.

12. Si vous souhaitez que MySQL se lance automatiquement au dèmarrage de votre machine, vous pouvez copier `support-files/mysql.server` á l'endroit oú votre systéme stocke ses fichiers de dèmarrage. Davantage d'informations sont disponibles dans le script `support-files/mysql.server` lui-même et dans Section 2.4.3 [Automatic start], page 104.

Aprés que tout soit installè, vous devez initialiser et tester votre distribution :

```
shell> /usr/local/mysql/bin/safe_mysqld --user=mysql &
```

Procèdez maintenant á Section 4.7.2 [`safe_mysqld`], page 292, et á la Voir Section 2.4 [Post-installation], page 97.

2.3 Installer MySQL á partir des sources

Avant de procèder á l'installation á partir des sources, vèrifiez auparavant que notre distribution binaire pour votre plate-forme ne fonctionne pas. Nous faisons un maximum d'efforts pour nous assurer que nos binaires sont compilès avec les meilleures options possibles.

Vous avez besoin des outils suivants pour compiler et installer MySQL á partir des sources :

- GNU `gunzip` pour dècompresser la distribution.
- Un programme `tar` pour dèsarchiver la distribution. GNU `tar` est connu pour fonctionner. Le `tar` de Sun connaît quelques problémes.
- Un compilateur C++ ANSI fonctionnel. `gcc` >= 2.95.2, `egcs` >= 1.0.2 ou `egcs` 2.91.66, SGI C++, et SunPro C++ sont quelques-uns des compilateurs rèputès pour fonctionner. `libg++` n'est pas nècessaire si vous utilisez `gcc`. `gcc` 2.7.x souffre d'un bogue qui l'empêche de compiler quelques fichiers C++ correctement ècrits, tels que '`sql/sql_base.cc`'. Si vous disposez seulement de `gcc` 2.7.x, vous devez mettre á

jour votre `gcc` afin de compiler MySQL. `gcc` 2.8.1 est aussi reconnu pour rencontrer des problémes sur certaines plate-formes, il devrait donc être dèsactivè si un autre compilateur existe pour la plate-forme.

`gcc` >= 2.95.2 est recommandè pour compiler MySQL dans ses versions 3.23.x.

- Un bon programme `make`. GNU `make` est une fois de plus recommandè et est quelquefois requis. Si vous rencontrez des problémes, nous vous recommandons d'essayer GNU `make` 3.75 ou supèrieur.

Si vous utilisez une version rècente de `gcc`, suffisamment rècente pour reconnaître l'option `-fno-exceptions`, il est **trés important** que vous l'utilisiez. Sinon, vous risquez de compiler un binaire qui crashe alèatoirement. Nous recommandons donc l'utilisation de `-felide-constructors` et `-fno-rtti` en même temps que `-fno-exceptions`. En cas de doute, faites la chose suivante :

```
CFLAGS="-O3" CXX=gcc CXXFLAGS="-O3 -felide-constructors -fno-exceptions \
       -fno-rtti" ./configure --prefix=/usr/local/mysql --enable-assembler \
       --with-mysqld-ldflags=-all-static
```

Sur la plupart des systémes, il en rèsultera un binaire rapide et stable.

Si vous rencontrez des problémes, **utilisez toujours** `mysqlbug` pour poster des questions á `mysql@lists.mysql.com`. Même si le probléme n'est pas un bogue, `mysqlbug` rassemble des informations sur le systéme qui aidera les autres á rèsoudre votre probléme. En n'utilisant pas `mysqlbug`, vous amoindrissez vos chances d'obtenir une solution á votre probléme ! Vous trouverez `mysqlbug` dans le rèpertoire '`scripts`' aprés avoir dèsarchivè la distribution. Voir Section 1.6.2.3 [Bug reports], page 27.

2.3.1 Vue d'ensemble de l'installation rapide

Les commandes de base que vous devez exècuter pour installer MySQL á partir des sources sont :

```
shell> groupadd mysql
shell> useradd -g mysql mysql
shell> gunzip < mysql-VERSION.tar.gz | tar -xvf -
shell> cd mysql-VERSION
shell> ./configure --prefix=/usr/local/mysql
shell> make
shell> make install
shell> scripts/mysql_install_db
shell> chown -R root  /usr/local/mysql
shell> chown -R mysql /usr/local/mysql/var
shell> chgrp -R mysql /usr/local/mysql
shell> cp support-files/my-medium.cnf /etc/my.cnf
shell> /usr/local/mysql/bin/safe_mysqld --user=mysql &
ou
shell> /usr/local/mysql/bin/mysqld_safe --user=mysql &
si vous utilisez MySQL 4.x.
```

Si vous voulez avoir le support des tables InnoDB, vous devez èditer le fichier `/etc/my.cnf` et enlever le caractére `#` avant le paramétre qui commence par `innodb_...`. Voir Section 4.1.2 [Option files], page 198, et Section 7.5.2 [InnoDB start], page 545.

Si vous utilisez un fichier RPM source, faites :

```
shell> rpm --rebuild MySQL-VERSION.src.rpm
```

Cela produira un fichier RPM binaire que vous pourrez installer.

Vous pouvez ajouter des utilisateurs en lançant le script `bin/mysql_setpermission` si vous installez les modules Perl `DBI` et `Msql-Mysql-modules`.

Ci-dessous, une description plus dètaillèe.

Pour installer MySQL á partir des sources, suivez ces ètapes et procèdez á Section 2.4 [Post-installation], page 97, pour la configuration et les tests consècutifs á l'installation :

1. Notez le rèpertoire oú vous voulez dècompacter les sources et dèplacez vous y.
2. Rècupèrez une distribution des sources á partir d'un des sites listès sur Section 2.2.2 [Getting MySQL], page 73.
3. Si vous souhaitez utiliser les tables Berkeley DB avec MySQL, vous devez obtenir une version modifièe du code source de Berkeley DB. Veuillez lire le chapitre relatif aux tables Berkeley DB avant de procèder. Voir Section 7.6 [BDB], page 585.

   Les distributions des sources MySQL sont fournies sous forme d'archive `tar` compressèes et ont des noms comme '`mysql-VERSION.tar.gz`', oú `VERSION` est un nombre comme 4.1.1-alpha.
4. Ajoutez un utilisateur et un groupe avec les droits desquels `mysqld` fonctionnera :

   ```
   shell> groupadd mysql
   shell> useradd -g mysql mysql
   ```

   Ces commandes ajoutent le groupe `mysql` group et l'utilisateur `mysql`. La syntaxe de `useradd` et de `groupadd` peut varier de façon significative suivant les versions d'Unix. Elles peuvent aussi s'appeler `adduser` et `addgroup`. Vous pouvez aussi souhaiter nommer le groupe et l'utilisateur autrement que `mysql`.
5. Dècompressez la distribution dans le rèpertoire courant :

   ```
   shell> gunzip < /path/to/mysql-VERSION.tar.gz | tar xvf -
   ```

   Cette commande crèe un rèpertoire nommè '`mysql-VERSION`'.
6. Dèplacez-vous dans le rèpertoire racine de la distrbution dècompressèe :

   ```
   shell> cd mysql-VERSION
   ```

   Notez bien que vous devez alors configurer et compiler MySQL depuis ce rèpertoire racine. Vous ne pouvez pas le compiler á partir d'un autre rèpertoire.
7. Configurez votre version et compilez tout :

   ```
   shell> ./configure --prefix=/usr/local/mysql
   shell> make
   ```

   Quand vous lancez la commande `configure`, vous pouvez spècifier quelques options. Lancez `./configure --help` pour une liste des options disponibles. La partie Section 2.3.3 [les options de `configure`], page 88, traite des options les plus utiles.

   Si `configure` èchoue, et que vous êtes sur le point d'envoyer un mail á `mysql@lists.mysql.com` pour demander de l'aide, ajoutez s'il vous plait les quelques

lignes de ‘`config.log`’ qui pourraient selon vous aider á rèsoudre le probléme. Ajoutez aussi les deux derniéres lignes de sortie de `configure` si `configure` s’arrête prècipitemment. Envoyez le rapport de bogue en utilisant le script `mysqlbug`. Voir Section 1.6.2.3 [Bug reports], page 27.

Si la compilation èchoue, rèfèrez-vous á Section 2.3.5 [Compilation problems], page 92 pour de l’aide sur les problémes les plus courants.

8. Installez tout :

```
shell> make install
```

Vous aurez certainement besoin de lancer cette commande en tant que `root`.

9. Crèez les tablees de gestion des droits MySQL (ceci est nècessaire uniquement si vous n’avez pas installè MySQL auparavant):

```
shell> scripts/mysql_install_db
```

Veuillez noter que les versions de MySQL antèrieures á la version 3.22.10 dèmarraient le serveur MySQL quand vous exècutiez `mysql_install_db`. Ce n’est plus le cas!

10. Nommez `root` en tant que propriètaire des binaires et l’utilisateur avec les droits duquel vous ferez fontionner `mysqld` comme propriètaire du rèpertoire de donnèes :

```
shell> chown -R root  /usr/local/mysql
shell> chown -R mysql /usr/local/mysql/var
shell> chgrp -R mysql /usr/local/mysql
```

La premiére commande change les attributs `owner` en faveur de l’utilisateur `root`, la seconde change les attributs `owner` du rèpertoire de donnèes en faveur de l’utilisateur `mysql`, et la troisiéme change les attributs `group` en faveur du groupe `mysql`.

11. Si vous voulez installer le support de l’interface Perl `DBI/DBD`, rèfèrez-vous á Section 2.7 [Perl support], page 149.
12. Si vous souhaitez que MySQL se lance automatiquement au dèmarrage de votre machine, vous pouvez copier `support-files/mysql.server` á l’endroit oú votre systéme stocke ses fichiers de dèmarrage. Davantage d’informations sont disponibles dans le script `support-files/mysql.server` lui-même et dans Section 2.4.3 [Automatic start], page 104.

Aprés que tout soit installè, vous devez initialiser et tester votre distribution :

```
shell> /usr/local/mysql/bin/safe_mysqld --user=mysql &
```

Si cette commande èchoue immèdiatement sur un `mysqld daemon ended`, vous pouvez trouvez des informations dans le fichier ‘`mysql-data-directory/'hostname'.err`’. La raison la plus courante est que vous avez dèjá un autre serveur `mysqld` qui fonctionne. Voir Section 4.1.4 [Multiple servers], page 202.

Procèdez maintenant á la Section 2.4 [Post-installation], page 97.

2.3.2 Appliquer des patches

Parfois, des patches sont publiès sur la liste de diffusion ou sont placès sur la page des patches sur le site de MySQL (`http://www.mysql.com/Downloads/Patches/`).

Pour appliquer un patch venant de la liste de diffusion, sauvegardez le message qui contient le patch, placez-le dans le rèpertoire racine de vos sources MySQL et lancez les commandes suivantes :

```
shell> patch -p1 < patch-file-name
shell> rm config.cache
shell> make clean
```

Les patches venant du site FTP sont distribuès sous forme de fichiers en texte clair ou bien sous forme de fichiers compressès avec `gzip`. Appliquez un patch en texte clair de la même façon qu'avec les patches de la liste de diffusion. Pour appliquer un patch compressè, dèplacez-vous dans le rèpertoire racine de vos sources et lancez les commandes suivantes :

```
shell> gunzip < patch-file-name.gz | patch -p1
shell> rm config.cache
shell> make clean
```

Aprés avoir appliquè un patch, suivez les instructions d'une installation normale á partir des sources, á commencer par l'ètape du `./configure`. Aprés avoir lancè le `make install`, redèmarrez votre serveur MySQL.

Vous aurez certainement besoin d'arrêter tout autre serveur MySQL avant de lancer `make install`. (Utilisez pour cela `mysqladmin shutdown`.) Certains systémes n'autorisent pas l'installation d'une nouvelle version d'un programme ; elle remplace la version qui est en train de s'exècuter.

2.3.3 Options habituelles de `configure`

Le script `configure` vous donne un bon moyen de contrôler la configuration de votre distribution MySQL. Habituellement, vous faites cela en spècifiant les options dans la ligne de commande de `configure`. Vous pouvez aussi affecter le comportement de `configure` en utilisant certaines variables d'environnement. Voir Annexe F [Environment variables], page 828. Pour une liste des options supportèes par `configure`, exècutez cette commande :

```
shell> ./configure --help
```

Les options de `configure` les plus utilisèes sont dècrites ici :

- Pour ne compiler que les librairies et programmes clients, et non le serveur, utilisez l'option `--without-server` :

  ```
  shell> ./configure --without-server
  ```

  Si vous n'avez pas de compilateur C++, `mysql` ne compilera pas (c'est le programme client qui requiére C++). Dans ce cas, vous pouvez supprimer la partie de code dans `configure` qui vèrifie l'existence d'un compilateur C++, puis exècuter `./configure` avec l'option `--without-server`. La compilation essaiera encore de construire `mysql`, mais vous pouvez ignorer les messages d'erreurs concernant '`mysql.cc`'. (Si `make` stoppe, essayez `make -k` pour dire de continuer même si on rencontre des erreurs.)
- Si vous voulez obtenir une librairie MySQL embarquèe (`libmysqld.a`) vous devez utiliser l'option `--with-embedded-server`.
- Si vous ne voulez pas que vos fichiers de log et bases de donnèes soient dans '`/usr/local/var`', utiliser une commande `configure` se rapprochant de l'une des commandes suivantes :

  ```
  shell> ./configure --prefix=/usr/local/mysql
  shell> ./configure --prefix=/usr/local \
             --localstatedir=/usr/local/mysql/data
  ```

La premiére change le prèfixe de l'installation pour que tout soit installè dans '`/usr/local/mysql`' au lieu de '`/usr/local`' par dèfaut. La seconde commande prèserve le prèfixe d'installation par dèfaut mais change le rèpertoire par dèfaut pour les bases de donnèes (normalement '`/usr/local/var`') en `/usr/local/mysql/data`. Aprés que vous ayez compilè MySQL, vous pouvez changer ces options dans les fichiers d'options. Voir Section 4.1.2 [Option files], page 198.

- Si vous utilisez Unix et que vous voulez que la socket de MySQL soit á un autre endroit que celui par dèfaut (normalement '`/tmp`' ou '`/var/run`') utilisez une commande `configure` comme celle-ci :

```
shell> ./configure --with-unix-socket-path=/usr/local/mysql/tmp/mysql.sock
```

  Notez que le fichier donnè doit avoir un chemin absolu ! Vous pourrez aussi changer le chemin vers '`mysql.sock`' plus tard en utilisant les fichiers d'options de MySQL. Voir Section A.4.5 [Problems with mysql.sock], page 702.

- Si vous voulez compiler des programmes liès statiquement (par exemple, pour crèer une distribution binaire, pour obtenir plus de vitesse, ou pour rèsoudre des problémes avec quelques distributions RedHat Linux), exècutez `configure` de la maniére suivante :

```
shell> ./configure --with-client-ldflags=-all-static \
           --with-mysqld-ldflags=-all-static
```

- Si vous utilisez `gcc` et n'avez pas `libg++` ou `libstdc++` d'installès, vous pouvez dire á `configure` d'utiliser `gcc` en tant que compilateur C++ :

```
shell> CC=gcc CXX=gcc ./configure
```

  Quand vous utilisez `gcc` en tant que compilateur C++, aucune tentative de liaison avec `libg++` ou `libstdc++` ne sera effectuèe. Il peut être bon d'utiliser cette mèthode même si vous avez les librairies citèes, car quelques versions de celles-ci ont causè des problémes á des utilisateurs MySQL par le passè.

  Voici quelques variables d'environnement á dèfinir selon le compilateur que vous utilisez :

Compilateur	**Options recommandèes**
gcc 2.7.2.1	CC=gcc CXX=gcc CXXFLAGS="-O3 -felide-constructors"
egcs 1.0.3a	CC=gcc CXX=gcc CXXFLAGS="-O3 -felide-constructors -fno-exceptions -fno-rtti"
gcc 2.95.2	CFLAGS="-O3 -mpentiumpro" CXX=gcc CXXFLAGS="-O3 -mpentiumpro \ -felide-constructors -fno-exceptions -fno-rtti"
pgcc 2.90.29 ou plus rècent	CFLAGS="-O3 -mpentiumpro -mstack-align-double" CXX=gcc \ CXXFLAGS="-O3 -mpentiumpro -mstack-align-double -felide-constructors \ -fno-exceptions -fno-rtti"

  Dans la plupart des cas, vous pouvez obtenir un binaire MySQL raisonnablement optimal en utilisant les options de la table prècèdente et en ajoutant les options suivantes aux lignes de configuration :

```
--prefix=/usr/local/mysql --enable-assembler \
--with-mysqld-ldflags=-all-static
```

  En d'autres termes, la ligne de configuration ressemble á ce qui suit pour les versions rècentes de gcc :

```
CFLAGS="-O3 -mpentiumpro" CXX=gcc CXXFLAGS="-O3 -mpentiumpro \
```

```
-felide-constructors -fno-exceptions -fno-rtti" ./configure \
--prefix=/usr/local/mysql --enable-assembler \
--with-mysqld-ldflags=-all-static
```

Les binaires que nous fournissons sur le site de MySQL á `http://www.mysql.com/` sont tous compilès avec une optimisation totale et devraient être parfaits pour la plupart des utilisateurs. Voir Section 2.2.7 [MySQL binaries], page 80. Il y'a quelques choses que vous pouvez modifier pour rendre le binaire encore plus rapide, mais cela est rèservè aux utilisateurs avancès. Voir Section 5.5.3 [Compile and link options], page 392.

Si la gènèration èchoue et produit des erreurs disant que votre compilateur ou outil de liaison n'est pas capable de crèer la librairie partagèe '`libmysqlclient.so.#`' ('`#`' ètant un numèro de version), vous pouvez contourner ce probléme en donnant l'option `--disable-shared` á `configure`. Dans ce cas, `configure` ne gènèrera pas de librairie partagèe '`libmysqlclient.so.#`'.

- Vous pouvez configurer MySQL pour qu'il n'utilise pas les valeurs `DEFAULT` pour les colonnes non-`NULL` (c'est á dire, les colonnes qui ne peuvent être `NULL`). Cela fait gènèrer des erreurs aux requêtes `INSERT` si vous ne spècifiez pas explicitement une valeur pour toutes les colonnes non-`NULL`. Pour supprimer l'utilisation des valeurs par dèfaut, exècutez `configure` de cette façon :

  ```
  shell> CXXFLAGS=-DDONT_USE_DEFAULT_FIELDS ./configure
  ```

- Par dèfaut, MySQL utilise le jeu de caractéres ISO-8859-1 (Latin1). Pour changer le jeu par dèfaut, utilisez l'option `--with-charset` :

  ```
  shell> ./configure --with-charset=CHARSET
  ```

  `CHARSET` peut être l'un des `big5`, `cp1251`, `cp1257`, `czech`, `danish`, `dec8`, `dos`, `euc_kr`, `gb2312`, `gbk`, `german1`, `hebrew`, `hp8`, `hungarian`, `koi8_ru`, `koi8_ukr`, `latin1`, `latin2`, `sjis`, `swe7`, `tis620`, `ujis`, `usa7`, ou `win1251ukr`. Voir Section 4.6.1 [Character sets], page 286.

  Si vous voulez convertir les caractéres entre le serveur et le client, regardez du côtè de la commande `SET CHARACTER SET`. Voir Section 5.5.6 [`SET`], page 396.

  Attention : Si vous changez les jeux de caractéres aprés avoir crèe des tables, vous devrez exècuter `myisamchk -r -q --set-character-set=charset` sur chaque table. Vous index pourraient être stockès de maniére incorrecte sinon. (Cela peut survenir si vous installez MySQL, crèez quelques tables, puis reconfigurez MySQL pour qu'il utilise un jeu de caractéres diffèrent et le rèinstallez.)

  Avec l'option `--with-extra-charsets=LIST` vous pouvez dèfinir les jeux de caractéres additionnels á compiler dans le serveur.

  Ici `LIST` est soit une liste de jeux de caractéres sèparès par des espaces, soit `complex` pour inclure tous les jeux de caractéres ne pouvant être chargès dynamiquement, ou encore `all` pour inclure tous les jeux de caractéres dans les binaires.

- Pour configurer MySQL avec le code de dèbogage, utilisez l'option `--with-debug` :

  ```
  shell> ./configure --with-debug
  ```

  Cela alloue un vèrificateur d'allocation de mèmoire qui peut trouver quelques erreurs et qui fournit des informations sur ce qui se produit. Voir Section E.1 [Debugging server], page 815.

- Si vos programmes clients utilisent les threads, vous avez besoin de compiler une version sûre pour les threads de la librairie du client MySQL avec l'option de configuration `--enable-thread-safe-client`. Cela crèera une librairie `libmysqlclient_r` avec laquelle vous devez lier vos applications threadèes. Voir Section 8.4.8 [Threaded clients], page 660.
- Les options relatives á un systéme d'exploitation particulier peuvent être trouvèes dans la section spècifique aux systémes de ce manuel. Voir Section 2.6 [Operating System Specific Notes], page 113.

2.3.4 Installer á partir de l'arbre source de dèveloppement

Attention : Vous devez lire cette partie seulement si vous voulez nous aider á tester notre nouveau code. Si vous souhaitez seulement faire fonctionner MySQL sur votre systéme, vous devriez utiliser la distribution d'une version standard (que ce soit une distribution sous forme de sources ou de binaire).

Pour obtenir notre arbre source de dèveloppement le plus rèent, suivez les instructions suivantes :

1. Tèlèchargez `BitKeeper` á partir de `http://www.bitmover.com/cgi-bin/download.cgi`. Vous aurez besoin de `Bitkeeper` 2.0 ou supèrieur pour accèder á notre dèpôt.
2. Suivez les instructions pour l'installer.
3. Aprés avoir installè `BitKeeper`, commencez par vous dèplacer dans le rèpertoire á partir duquel vous voulez travailler, et lancez l'une des commandes suivantes pour dupliquer la branche MySQL de votre choix :

   Pour dupliquer la branche 3.23, utilisez cette commande :

   ```
   shell> bk clone bk://work.mysql.com:7000 mysql-3.23
   ```

   Pour dupliquer la branche 4.0, utilisez cette commande :

   ```
   shell> bk clone bk://work.mysql.com:7001 mysql-4.0
   ```

   Pour dupliquer la branche 4.1, utilisez cette commande :

   ```
   shell> bk clone bk://work.mysql.com:7004 mysql-4.1
   ```

   Dans les exemples prècèdents, l'arbre source sera dèposè dans les sous-rèpertoires '`mysql-3.23/`', '`mysql-4.0/`', ou '`mysql-4.1/`' de votre rèpertoire courant.

   Le premier tèlèchargement de l'arbre source peut prendre un certain temps, selon la vitesse de votre connexion. Soyez patients.
4. Vous aurez besoin de GNU `autoconf 2.52`, de `automake 1.5`, de `libtool 1.4`, et de `m4` pour lancer la prochaine sèrie de commandes.

   ```
   shell> cd mysql-4.0
   shell> bk -r get -Sq
   shell> aclocal; autoheader; autoconf;  automake;
   shell> ./configure  # Ajoutez ici vos options favorites
   shell> make
   ```

   Si vous obtenez des erreurs ètranges pendant cette ètape, vèrifiez bien que vous avez vraiment installè `libtool`!

Une collection de nos scripts de configuration les plus courants de trouve dans le sous-rèpertoire '`BUILD/`'. Si vous êtes fainèants, vous pouvez utiliser '`BUILD/compile-pentium-debug`'. Pour compiler sur une architecture diffèrente, modifiez ce script en enlevant les drapeaux spècifiques au Pentium.

5. Quand la compilation est achevèe, lancez `make install`. Prenez garde sur des machines de production. Cette commande pourrait ècraser votre installation actuelle. Si vous avez une autre installation de MySQL, nous vous recommandons de lancer `./configure` avec des valeurs des options `prefix`, `with-tcp-port`, et `unix-socket-path` diffèrentes de celles de votre serveur de production.
6. Torturez votre nouvelle installation et tentez de faire planter les nouvelles fonctionnalitès. Commencez par lancer `make test`. Voir Section 9.1.2 [MySQL test suite], page 670.
7. Si vous avez èchouè avec l'ètape `make` et que la distribution ne compile pas, envoyez un rapport de l'incident á `bugs@lists.mysql.com`. Si vous avez installè la derniére version des indispensables outils GNU, et qu'ils èchouent dans l'analyse de vos fichiers de configuration, envoyez aussi un rapport. D'autre part, si vous exècutez `aclocal` et que vous obtenez l'erreur `command not found` ou un probléme du même type, n'envoyez pas de rapport. A la place, assurez vous que les outils nècessaires sont bien installès et que votre variable `PATH` est configurèe de telle façon que votre interprèteur de commandes les trouvent.
8. Aprés la premiére opèration `bk clone` pour obtenir l'arbre source, vous devez lancer règuliérement `bk pull` pour obtenir les mises á jour.
9. Vous pouvez examiner l'historique des changements de l'arbre avec toutes les diffèrences en utilisant `bk sccstool`. Si vous apercevez des diffèrences anormales ou sur lesquelles vous avez des questions, n'hèsitez pas á envoyer un e-mail á `internals@lists.mysql.com`. De même, si vous pensez avoir une meilleure mèthode pour traiter un probléme, envoyez un e-mail accompagnè d'un patch á la même adresse. `bk diffs` vous fournira un patch aprés que vous ayez fait vos changements aux sources. Si vous n'avez pas le temps de coder votre idèe, envoyez en juste une description.
10. `BitKeeper` dispose d'une aide agrèable á laquelle vous pouvez accèder via `bk helptool`.
11. Veuillez noter que chaque commit (`bk ci` ou `bk citool`) postera un message avec un aperçu des changements á notre liste de diffusion interne, á la façon habituelle des propositions openlogging.org avec seulement les commentaires des changements. Gènèralement, vous n'aurez pas besoin d'utiliser commit (l'arbre public interdisant les `bk push`), mais plutôt d'utiliser la mèthode `bk diffs` dècrite plus haut.

2.3.5 Problémes de compilation?

Tous les programmes MySQL compilent proprement chez nous, sans aucune alerte sur Solaris avec `gcc`. Sur d'autres systémes, des alertes peuvent apparaître á cause de diffèrences dans le systéme d'inclusions. Voyez Section 2.3.6 [MIT-pthreads], page 95 pour les alertes qui peuvent apparaître avec MIT-pthreads. Pour d'autres problémes, voyez la liste suivante.

La solution á de nombreux problémes implique une reconfiguration. Si vous avez besoin de faire une reconfiguration voici quelques conseils gènèraux :

- Si `configure` est exècutè aprés une premiére exècution, il peut utiliser des informations qui ont ètè rassemblèes durant une premiére invocation. Ces informations sont stockèes dans le fichier '`config.cache`'. Lorsque `configure` est lancè, il commence par regarder dans ce fichier, et lire le contenu qui existe, en supposant que ces donnèes sont toujours correctes. Cette supposition est invalide si vous faîtes une reconfiguration.
- Chaque fois que vous exècutez `configure`, vous devez exècuter á nouveau `make` pour recompiler. Toutefois, vous devrez peut être supprimer les vieux fichiers d'objets qui ont ètè compilè en utilisant diffèrentes configurations prècèdentes.

Pour èviter d'utiliser de vieilles informations de configuration, ou des vieux fichiers d'objet, vous pouvez utiliser ces commandes, avant `configure` :

```
shell> rm config.cache
shell> make clean
```

Alternativement, vous pouvez aussi utiliser `make distclean`.

La liste suivante dècrit certains problémes lors de la compilation de MySQL, qui surviennent souvent :

- Si vous avez des problémes lors de la compilation de '`sql_yacc.cc`', comme ceux qui sont dècrits ci-dessous, vous avez probablement ètè á court de mèmoire ou d'espace de swap :

  ```
  Internal compiler error: program cc1plus got fatal signal 11
    ou
  Out of virtual memory
    ou
  Virtual memory exhausted
  ```

  Le probléme est que `gcc` requiert de grandes quantitè de mèmoire pour compiler '`sql_yacc.cc`' avec les fonctions inline. Essayez d'exècuter `configure` avec l'option `--with-low-memory` :

  ```
  shell> ./configure --with-low-memory
  ```

  Cette option ajoute `-fno-inline` dans la ligne de compilation, si vous utilisez `gcc` et `-O0` si vous utilisez autre chose. Vous pouvez essayer `--with-low-memory` même si il vous reste suffisamment de mèmoire, et que vous ne pensez pas être limitè. Ce probléme a ètè observè sur des systémes avec de gènèreuses configurations, et `--with-low-memory` rèsout ce probléme.
- Par dèfaut, `configure` choisit `c++` comme compilateur, et GNU `c++` pour les liens avec `-lg++`. Si vous utilisez `gcc`, ce comportement peut poser les problémes suivants :

  ```
  configure: error: installation or configuration problem:
  C++ compiler cannot create executables.
  ```

  Vous pourriez aussi observer des problémes durant la compilation, avec `g++`, `libg++` ou `libstdc++`.

  La cause de ces problémes est que vous avez peut être que vous n'avez pas `g++`, ou que vous avez `g++` mais pas `libg++`, ou `libstdc++`. Regardez le fichier de log '`config.log`'. Il va sûrement contenir la raison exacte du mauvais fonctionnement de votre compilateur. Pour contourner ce probléme, vous pouvez utiliser `gcc` comme compilateur C++. Essayez de modifier la variable d'environnement `CXX` avec la valeur `"gcc -O3"`. Par exemple :

```
shell> CXX="gcc -O3" ./configure
```

Cela fonctionne car `gcc` compile les sources C++ aussi bien que `g++`, mais il n'est pas liè avec `libg++` ou `libstdc++` par dèfaut.

Un autre moyen pour règler ces problémes, bien sur, est d'installer `g++`, `libg++` et `libstdc++`. Nous vous recommandons toutefois de ne pas utiliser `libg++` ou `libstdc++` avec MySQL car cela ne fera qu'accroître la taille de votre exècutable binaire, sans vous apporter d'avantages. Par le passè, certaines versions de ces librairies ont posè des problémes ètranges aux utilisateurs MySQL.

- Si votre compilation èchoue avec des erreurs, ou si l'une des erreurs suivantes apparaît, vous devez changer la version de `make` en GNU `make`:

```
making all in mit-pthreads
make: Fatal error in reader: Makefile, line 18:
Badly formed macro assignment
```

ou

```
make: file 'Makefile' line 18: Must be a separator (:
```

ou

```
pthread.h: No such file or directory
```

Solaris et FreeBSD sont connus pour avoir des problémes avec `make`.

GNU `make` version 3.75 est reconnu pour fonctionner.

- Si vous voulez dèfinir des options supplèmentaires qui seront utilisèes par votre compilateur C ou C++, faites le en ajoutant ces options aux variables d'environnement `CFLAGS` et `CXXFLAGS`. Vous pouvez aussi spècifier le nom du compilateur via les variables `CC` et `CXX`. Par exemple :

```
shell> CC=gcc
shell> CFLAGS=-O3
shell> CXX=gcc
shell> CXXFLAGS=-O3
shell> export CC CFLAGS CXX CXXFLAGS
```

Voyez Section 2.2.7 [MySQL binaries], page 80, pour avoir une liste des dèfinitions des options disponibles sur divers systémes.

- Si vous obtenez un message d'erreur comme celui-ci, vous devrez mettre á jour votre version de `gcc` :

```
client/libmysql.c:273: parse error before '__attribute__'
```

`gcc` 2.8.1 est connu pour fonctionner, mais nous recommandons l'utilisation de `gcc` 2.95.2 ou `egcs` 1.0.3a.

- Si vous obtenez des erreurs telles que celles qui sont affichèes ci-dessous lors de la compilation de `mysqld`, c'est que `configure` n'a pas correctement dètecter le dernier argument des fonctions `accept()`, `getsockname()` ou `getpeername()` :

```
cxx: Error: mysqld.cc, line 645: In this statement, the referenced
     type of the pointer value "&length" is "unsigned long", which
     is not compatible with "int".
new_sock = accept(sock, (struct sockaddr *)&cAddr, &length);
```

Pour corriger ce probléme, èditez le fichier '`config.h`' (qui est gènèrè par le fichier `configure`). Recherchez ces lignes :

```
/* Define as the base type of the last arg to accept */
#define SOCKET_SIZE_TYPE XXX
```

Remplacez `XXX` par `size_t` ou `int`, suivant votre systéme d'exploitation. Notez que vous devrez faire cette manipulation á chaque fois que vous exècuterez le script `configure` car `configure` regènére '`config.h`'.

- Le fichier '`sql_yacc.cc`' est gènèrè á partir du fichier '`sql_yacc.yy`'. Normalement, le processus de crèation ne s'occupe pas de '`sql_yacc.cc`', car MySQL en a dèjá une copie. Cependant, si vous devez le recrèer, vous pouvez rencontrer cette erreur :

  ```
  "sql_yacc.yy", line xxx fatal: default action causes potential...
  ```

  Cela indique que votre version de `yacc` est inadèquate. Vous devrez probablement rèinstaller `bison` (la version GNU de `yacc`) et l'utiliser á la place.
- Si vous avez besoin de dèboguer `mysqld` ou un client MySQL, exècutez le script `configure` avec l'option `--with-debug`, puis recompilez vos clients avec la nouvelle librairie. Voir Section E.2 [Debugging client], page 821.

2.3.6 Notes relatives aux MIT-pthreads

Cette section dècrit quelques informations concernant l'utilisation des MIT-pthreads.

Notez que sur Linux vous **ne** devez **pas** utiliser les MIT-pthreads mais installer LinuxThreads ! Voir Section 2.6.1 [Linux], page 113.

Si votre systéme ne fournit pas un support nattif des threads, vous aurez besoin de construire MySQL en utilisant le package des MIT-pthreads. Cela inclut les anciens systémes FreeBSD, SunOS 4.x, Solaris 2.4 et plus ancien, et quelques autres systémes. Voir Section 2.2.3 [Which OS], page 74.

Notez qu'á partir de la version 4.0.2 de MySQL les MIT-pthreads ne font plus partie de la distribution des sources ! si vous avez besoin de ce package, vous pouvez l'obtenir sur `http://www.mysql.com/Downloads/Contrib/pthreads-1_60_beta6-mysql.tar.gz`

Aprés l'avoir rècupèrè, dècompressez l'archive dans le rèpertoire racine de votre rèpertoire des sources de MySQL. Cela crèera le rèpertoire `mit-pthreads`.

- Sur la plupart des systémes, vous pouvez forcer l'utilisation des MIT-pthreads en exècutant `configure` avec l'option `--with-mit-threads` :

  ```
  shell> ./configure --with-mit-threads
  ```

  La compilation dans un dossier non-sources n'est pas supportè lors de l'utilisation des MIT-pthreads car nous voulons minimiser les changements de leur code.
- La vèrification pour l'utilisation des MIT-pthreads ne survient que durant la partie du processus de configuration qui s'occupe du code du serveur. Si vous avez configurè la distribution en utilisant `--without-server` pour ne construire que le client, les clients ne sauront pas si les MIT-pthreads sont utilisès et utiliserons les socket Unix pour les connexions par dèfaut. Puisque les sockets Unix ne fonctionnent pas avec les MIT-pthreads sur certaines plate-formes, cela signifie que vous devrez utiliser `-h` ou `--host` quand vous exècuterez les programmes clients.
- Lorsque MySQL est compilè en utilisant les MIT-pthreads, le verrouillage systéme est dèsactivè par dèfaut pour des soucis de performances. Vous pouvez demander au

serveur d'utiliser les verrous systémes avec l'option `--external-locking`. Cela n'est requis que si vous avez besoin de faire fonctionner deux serveurs MySQL avec les mêmes donnèes (non recommandè).

- De temps en temps, la commande `bind()` des pthreads n'arrive pas á attacher une socket sans afficher d'erreurs (du mois, sous Solaris). Le rèsultat est que toutes les connexions au serveur èchouent. Par exemple :

```
shell> mysqladmin version
mysqladmin: connect to server at '' failed;
error: 'Can't connect to mysql server on localhost (146)'
```

  La solution est de terminer le serveur `mysqld` et de le redèmarrer. Cela ne nous est arrivè que quand nous avons forcè le serveur á se terminer et que nous l'avons redèmarrè immèdiatement aprés.

- Avec les MIT-pthreads, l'appel systéme á `sleep()` n'est pas interruptible avec `SIGINT` (break). On ne s'en rend compte qua quand on exècute `mysqladmin --sleep`. Vous devez attendre que l'appel systéme á `sleep()` se termine avant que le processus ne s'arrête.

- Lors de la liaison, vous pouvez obtenir des messages d'erreurs comme ceux-ci (du moins sur Solaris); ils peuvent être ignorès :

```
ld: warning: symbol `_iob' has differing sizes:
    (file /my/local/pthreads/lib/libpthread.a(findfp.o) value=0x4;
file /usr/lib/libc.so value=0x140);
    /my/local/pthreads/lib/libpthread.a(findfp.o) definition taken
ld: warning: symbol `__iob' has differing sizes:
    (file /my/local/pthreads/lib/libpthread.a(findfp.o) value=0x4;
file /usr/lib/libc.so value=0x140);
    /my/local/pthreads/lib/libpthread.a(findfp.o) definition taken
```

- D'autres avertissements peuvent être ignorès :

```
implicit declaration of function `int strtoll(...)'
implicit declaration of function `int strtoul(...)'
```

- Nous n'avons pas rèussi á faire fonctionner `readline` avec les MIT-pthreads. (Cela n'est pas nècessaire, mais peut être utile á quelqu'un.)

2.3.7 La distribution source Windows

Vous aurez besoin de :

- Un compilateur VC++ 6.0 (mis á jour avec 4 ou 5 SP et le paquet Prè-processeur) Le paquet Prè-processeur est nècessaire pour l'assembleur de macros. D'avantage de dètails sur `http://msdn.microsoft.com/vstudio/sp/vs6sp5/faq.asp`.
- La distribution source MySQL pour Windows, qui peut être obtenue sur `http://www.mysql.com/downloads/`.

Compiler MySQL

1. Crèez un rèpertoire de travail (par exemple, workdir)
2. Dècompressez la distribution source dans ce rèpertoire.

3. Lancez le compilateur VC++ 6.0.
4. Dans le menu `Fichier`, sèlectionnez `Ouvrir un espace de travail`.
5. Ouvrez l'espace de travail '`mysql.dsw`' que vous trouverez dans le rèpertoire de travail.
6. Dans le menu `Gènèrer`, sèlectionnez le menu `Choisir la configuration active`.
7. Choisissez `mysqld - Win32 Debug` puis cliquez sur OK.
8. Appuyez sur `F7` pour commencer á compiler le serveur de dèbogage, les bibliothéques, et quelques applications clientes.
9. Quand la compilation est terminèe, copiez les bibliothéques et les exècutables dans un rèpertoire sèparè.
10. Compilez les versions que vous souhaitez de la même maniére.
11. Crèez le rèpertoire pour les outils MySQL : par exemple, '`c:\mysql`'
12. A partir du rèpertoire workdir, copiez les rèpertoires suivants dans c:\mysql :
    - Data
    - Docs
    - Share
13. Crèez le rèpertoire '`c:\mysql\bin`' et copier toutes les applications serveur et client compilèes auparavant.
14. Si vous le souhaitez, vous pouvez aussi crèer le rèpertoire '`lib`' et y copier les librairies compilèes auparavant.
15. Faites un nettoyage en utilisant Visual Studio.

Configurez et lancez le serveur de la même façon que la distribution binaire Windows. Voir Section 2.1.2.2 [Windows prepare environment], page 71.

2.4 Configuration aprés l'installation, et tests

Une fois que vous avez installè MySQL (que ce soit avec les sources ou avec la distribution binaire), vous devez initialiser les tables de droits, dèmarrer le serveur et vous assurer que tout fonctionne bien. Vous pouvez aussi vouloir configurer le dèmarrage et l'extinction automatique du serveur.

Normalement, vous installez les tables de droits et dèmarrez le serveur comme ceci, pour une distribution source :

```
shell> ./scripts/mysql_install_db
shell> cd mysql_installation_directory
shell> ./bin/safe_mysqld --user=mysql &
```

Pour une distribution binaire (pas un RPM ou un package), faites ceci :

```
shell> cd mysql_installation_directory
shell> ./bin/mysql_install_db
shell> ./bin/safe_mysqld --user=mysql &
```

Cela crèe la base `mysql` qui va contenir tous les droits, ainsi que la base test, que vous pouvez utiliser pour tester MySQL, ainsi que les droits pour l'utilisateur qui fait fonctionner `mysql_install_db` et un utilisateur `root` (sans mots de passe). Cela va aussi dèmarrer le serveur `mysqld`.

`mysql_install_db` ne va pas ècraser d'anciens droits installè, et il peut être utilisè en toutes circonstances. Si vous ne voulez pas de base `test`, vous pouvez la supprimer avec la commande `mysqladmin -u root drop test`.

Les tests sont plus facilement exècutè depuis le niveau supèrieur de la distribution MySQL. Pour la distribution binaire, c'est votre dossier d'installation (typiquement, '`/usr/local/mysql`'). Pour une distribution source, c'est le dossier principal de votre dossier source MySQL.

Dans les commandes prèsentèes dans cette section et dans les sous sections suivantes, `BINDIR` est le chemin jusqu'au programmes binaires de la distribution, comme `mysqladmin` et `safe_mysqld`. Pour une distribution binaire, il s'agit du dossier '`bin`' dans la distribution. Pour une distribution source, `BINDIR` vaut sûrement '`/usr/local/bin`', á moins que vous ne spècifiez un dossier d'installation diffèrent que '`/usr/local`' avec le script `configure`. `EXECDIR` est le chemin jusqu'au serveur `mysqld`. Pour une distribution binaire, c'est la même valeur que `BINDIR`. Pour une distribution source, `EXECDIR` est probablement '`/usr/local/libexec`'.

Les tests sont dècrits en dètails :

1. Si besoin, dèmarrez le serveur `mysqld` et modifiez les tables initiales de droits de MySQL qui contiennent les priviléges qui dèterminent les droits de connexion des utilisateurs au serveur. Cela se fait normalement avec le script `mysql_install_db` :

   ```
   shell> scripts/mysql_install_db
   ```

   Typiquement, `mysql_install_db` doit être exècutè uniquement lors de votre premiére installation de MySQL. Par consèquent, si vous faites èvoluer une installation, vous pouvez èviter cette ètape. Cependant, `mysql_install_db` est tout á fait sècuritaire, et vous ne modifiera pas des tables existants, alors dans le doute, vous pouvez toujours utiliser `mysql_install_db`.

   `mysql_install_db` crèe six tables (`user`, `db`, `host`, `tables_priv`, `columns_priv`, and `func`) dans la base `mysql` database. Une description des droits initiaux est proposè dans la section Section 4.3.4 [Default privileges], page 232. En bref, ces droits permettront au `root` MySQL de faire ce qu'il souhaite, et notamment, d'autoriser la crèation et l'utilisation de la base `test` ou dont le nom commence par `test_`.

   Si vous ne configurez pas les tables de droits, l'erreur suivante va apparaître dans le fichier de log, au dèmarrage du serveur :

   ```
   mysqld: Can't find file: 'host.frm'
   ```

   Cela peut aussi arriver avec la distribution binaire de MySQL, si vous ne dèmarrez pas MySQL exactement avec `./bin/safe_mysqld`! Voir Section 4.7.2 [`safe_mysqld`], page 292.

   Vous pouvez avoir besoin d'exècuter `mysql_install_db` en tant que `root`. Toutefois, si vous pouvez exècuter le serveur MySQL en tant qu'utilisateur avec peu de droits (non-root), tant que cet utilisateur a les droits de lecture et d'ècriture dans le dossier de donnèes. Les instructions pour faire fonctionner MySQL avec un autre utilisateur sont donnèes dans le chapitre Section A.3.2 [Changing MySQL user], page 696.

   Si vous avez des problémes avec `mysql_install_db`, voyez Section 2.4.1 [`mysql_install_db`], page 101.

   Il y des alternatives á l'utilisation de `mysql_install_db` tel qu'il est fourni dans la distribution MySQL :

- Vous pouvez èditer le script `mysql_install_db` avant de l'exècuter, pour modifier les droits initiaux qui sont implantès dans les tables de droits. C'est pratique si vous voulez installer MySQL sur de nombreuses machines, avec les mêmes droits. Dans ce cas, vous aurez probablement á ajouter quelques insertions `INSERT` dans les tables `mysql.user` et `mysql.db`!
- Si vous voulez modifier des èlèments dans la table de droits aprés les avoir installè, vous pouvez exècuter `mysql_install_db`, puis utiliser `mysql -u root mysql` pour vous connecter aux tables de droit en tant que `root` MySQL, et exècuter les commandes SQL nècessaires, directement dans les tables de droits.
- Il est possible de recrèer complétement les tables de droits une fois qu'elles ont ètè crèèes. Vous pouvez faire cela si vous avez dèjá installè les tables, mais que vous voulez les recrèer aprés èdition de `mysql_install_db`.

Pour plus d'informations sur ces alternatives, voyez Section 4.3.4 [Default privileges], page 232.

2. Dèmarrez le serveur MySQL comme ceci :

```
shell> cd mysql_installation_directory
shell> bin/safe_mysqld &
```

Si vous avez des problémes avec le dèmarrage du serveur, voyez Section 2.4.2 [Starting server], page 102.

3. Utilisez `mysqladmin` pour vèrifier que le serveur fonctionne. Les commandes suivantes fournissent un accés simple pour tester si le serveur fonctionne correctement :

```
shell> BINDIR/mysqladmin version
shell> BINDIR/mysqladmin variables
```

Le rèsultat de `mysqladmin version` varie suivant votre plate-forme et votre version de MySQL, mais il doit être similaire á ceci :

```
shell> BINDIR/mysqladmin version
mysqladmin  Ver 8.14 Distrib 3.23.32, for linux on i586
Copyright (C) 2000 MySQL AB & MySQL Finland AB & TCX DataKonsult AB
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL license.

Server version          3.23.32-debug
Protocol version        10
Connection              Localhost via Unix socket
TCP port                3306
UNIX socket             /tmp/mysql.sock
Uptime:                 16 sec

Threads: 1  Questions: 9  Slow queries: 0
Opens: 7  Flush tables: 2  Open tables: 0
Queries per second avg: 0.000
Memory in use: 132K  Max memory used: 16773K
```

Pour savoir ce que vous pouvez faire d'autre avec `BINDIR/mysqladmin`, appelez le avec l'option `--help`.

4. Vèrifiez que vous pouvez èteindre le serveur :

```
shell> BINDIR/mysqladmin -u root shutdown
```

5. Verifiez que vous pouvez redèmarrer le serveur. Faites cela avec `safe_mysqld` ou directement avec `mysqld`. Par exemple :

```
shell> BINDIR/safe_mysqld --log &
```

   Si `safe_mysqld` èchoue, essayez d'exècuter MySQL depuis le dossier d'installation (si vous n'y êtes pas dèjá). Si cela ne fonctionne pas, voyez Section 2.4.2 [Starting server], page 102.

6. Exècutez un test simple pour vèrifier que le serveur fonctionne. Le rèsultat devrait être similaire á ces lignes ci :

```
shell> BINDIR/mysqlshow
+-----------+
| Databases |
+-----------+
| mysql     |
+-----------+

shell> BINDIR/mysqlshow mysql
Database: mysql
+--------------+
|    Tables    |
+--------------+
| columns_priv |
| db           |
| func         |
| host         |
| tables_priv  |
| user         |
+--------------+

shell> BINDIR/mysql -e "SELECT host,db,user FROM db" mysql
+------+--------+------+
| host | db     | user |
+------+--------+------+
| %    | test   |      |
| %    | test_% |      |
+------+--------+------+
```

   Il y a aussi la suite de performances dans le dossier '`sql-bench`' (dans le dossier d'installation MySQL) que vous pouvez utiliser pour comparer les performances de MySQL avec d'autres plate-formes. Le dossier '`sql-bench/Results`' contient le rèsultat de nombreux tests avec diffèrentes bases et plate-forme. Pour exècuter tous les tests, exècutez cette commande :

```
shell> cd sql-bench
shell> run-all-tests
```

Si vous n'avez pas de dossier '`sql-bench`', vous utilisez probablement une distribution binaire au format RPM. Les distribution source RPM incluent le dossier de tests. Dans ce cas, vous devez d'abord installer la suite de test avant de l'utiliser. Depuis MySQL Version 3.22, il y a des fichiers de tests RPM, appelè '`mysql-bench-VERSION-i386.rpm`' qui contient les donnèes de tests et les donnèes.

Si vous avez une distribution source, vous pouvez aussi exècuter les tests dans le dossier '`tests`'. Par exemple, pour exècuter '`auto_increment.tst`', faites ceci :

```
shell> BINDIR/mysql -vvf test < ./tests/auto_increment.tst
```

Les rèsultats attendus sont prèsentès dans le fichier '`./tests/auto_increment.res`'.

2.4.1 Problémes d'exècution de `mysql_install_db`

Le but du script `mysql_install_db` est de gènèrer un nouveau systéme de droits pour MySQL. Il ne modifiera aucune autre donnèe! Il ne fera rien du tout si vous avez des tables de droits installèes.

Si vous voulez recrèer vos tables de droits, vous devez èteindre le serveur `mysqld`, si il fonctionnait, et faire ceci :

```
mv mysql-data-directory/mysql mysql-data-directory/mysql-old
mysql_install_db
```

Cette section liste les problémes que vous pourriez rencontrer lors de l'exècution du script `mysql_install_db` :

`mysql_install_db` n'installe pas les tables de droits

Vous rèalisez que `mysql_install_db` n'arrive pas á installer les tables de droits, et se termine sur ce message :

```
starting mysqld daemon with databases from XXXXXX
mysql daemon ended
```

Dans ce cas, examinez le fichier de log trés attentivement! Le fichier de log est situè dans le dossier '`XXXXXX`' indiquè dans le message d'erreur, et il indiquera pourquoi `mysqld` n'a pas dèmarrè. SI vous ne comprenez pas ce qui est arrivè, incluez le log dans votre message, lors de l'envoi du rapport de bugs avec `mysqlbug`! Voir Section 1.6.2.3 [Bug reports], page 27.

Un dèmon `mysqld` fonctionne dèjá

Dans ce cas, vous n'avez probablement pas exècutè `mysql_install_db` du tout. Vous avez exècutè `mysql_install_db` une fois, lorsque vous avez installè MySQL pour la premiére fois.

Installer un second dèmon `mysqld` n'est pas possible lorsque le premier fonctionne.

Cela arrive lorsque vous avez une installation MySQL prè-existantes, mais que vous voulez installer une autre version ailleurs (par exemple, pour faire des tests ou simplement pour avoir deux installations). Gènèralement, le probléme survient lorsque le second serveur est dèmarrè, et qu'il essaie d'utiliser les mêmes ports et sockets que le premier. Dans ce cas, vous recevez des message d'erreur tels que : `Can't start server: Bind on TCP/IP port: Address already in use` ou `Can't start server: Bind on unix socket...`. Voir Section 4.1.3 [Installing many servers], page 201.

You don't have write access to '/tmp'

Si vous n'avez pas les droits d'accés suffisant pour crèer un fichier de socket á l'endroit prèvu (dans '/tmp') ou les permissions pour crèer un fichier temporaire dans '/tmp,' vous allez avoir une erreur lors de l'utilisation de mysql_install_db ou avec mysqld.

Vous pouvez spècifier une socket diffèrente et un dossier temporaire diffèrent avec les options suivantes :

```
shell> TMPDIR=/some_tmp_dir/
shell> MYSQL_UNIX_PORT=/some_tmp_dir/mysqld.sock
shell> export TMPDIR MYSQL_UNIX_PORT
```

Voir Section A.4.5 [Problems with mysql.sock], page 702.

'some_tmp_dir' doit être un chemin vers un dossier dans lequel vous avez les permissions d'ècriture. Voir Annexe F [Environment variables], page 828.

Aprés cela, vous devriez être capable d'exècuter mysql_install_db et dèmarrer le serveur avec ces commandes :

```
shell> scripts/mysql_install_db
shell> BINDIR/safe_mysqld &
```

mysqld crashes immediately

Si vous utilisez RedHat version 5.0 avec une version de glibc plus ancienne que 2.0.7-5, assurez vous que vous avez installè les patchs glibc! Il y a beaucoup de littèrature á ce propos dans les archives de listes de diffusion. Des liens vers les archives sont disponibles á http://lists.mysql.com/. Voir aussi Section 2.6.1 [Linux], page 113.

Vous pouvez aussi dèmarrer mysqld manuellement avec l'option --skip-grant-tables puis ajouter les tables de droits avec Smysql:

```
shell> BINDIR/safe_mysqld --skip-grant-tables &
shell> BINDIR/mysql -u root mysql
```

Dans la base mysql, exècutez manuellement les commandes SQL disponibles dans mysql_install_db. Assurez vous que vous n'oubliez pas la commande mysqladmin flush-privileges ou mysqladmin reload pour dire au serveur de prendre en compte les tables de droits.

2.4.2 Problémes de dèmarrage du serveur MySQL

Si vous allez utiliser des tables qui supportent les transactions (InnoDB, BDB), vous devez commencer par crèer un fichier 'my.cnf' et y placer les options de dèmarrage de ces tables. Voir Chapitre 7 [Table types], page 531.

Gènèralement, vous dèmarrez le serveur mysqld de l'une de ces façons :

- En utilisant le script mysql.server. Ce script est utilisè á la base au moment du dèmarrage et de l'extinction du systéme, et il est dècrit avec plus de dètail dans le chapitre Section 2.4.3 [Automatic start], page 104.
- En appelant le script safe_mysqld, qui va dèterminer les options correctes pour mysqld puis utiliser ces options. Voir Section 4.7.2 [safe_mysqld], page 292.

- Pour Windows NT/2000/XP, voyez Section 2.6.2.2 [NT start], page 122.
- En appelant `mysqld` directement.

Lorsque le dèmon `mysqld` dèmarre, il change le dossier de travail par le dossier de donnèes. C'est lá qu'il doit trouver les fichiers de log, et le fichier pid (ID de processus), ainsi que les dossiers de bases.

Le chemin du dossier de donnèes est codè en dur lorsque la distribution est compilèe. Cependant, si `mysqld` cherche le dossier de donnèes ailleurs que lá ou il est vraiment, il ne va pas fonctionner correctement. Si vous avez des problémes avec les chemins, vous pouvez utiliser les options dont `mysqld` dispose pour vous permettre de modifier dynamiquement le chemin du dossier de donnèes : il suffit d'appeler `mysqld` avec l'option `--help`. Vous pouvez remplacer les valeurs par dèfaut en spècifiant les chemins corrects en ligne de commande avec `mysqld`. Ces options fonctionneront aussi avec `safe_mysqld`.

Normalement, vous devez appeler `mysqld` uniquement depuis le dossier d'installation de MySQL. Vous pouvez faire cela avec l'option `--basedir`. Vous pouvez aussi consulter l'affichage de `--help` pour vèrifier le changement des options de chemin (notez que `--help` **doit** être l'option finale d'une commande `mysqld`. Par exemple :

```
shell> EXECDIR/mysqld --basedir=/usr/local --help
```

Une fois que vous avez dèterminè le chemin que vous souhaitez, vous pouvez dèmarrer le serveur sans l'option `--help`.

Quelque soit la mèthode que vous utilisez pour dèmarrer le serveur, si elle èchoue, vèrifiez le fichier de log pour savoir pourquoi. Les fichiers de log sont situès dans le dossier de donnèes (typiquement '`/usr/local/mysql/data`' pour une distribution binaire, '`/usr/local/var`' pour une distribution source, et '`\mysql\data\mysql.err`' sous Windows). Regardez dans le dossier de donnèes et recherchez des fichiers de la forme '`host_name.err`' et '`host_name.log`' ou `host_name` est le nom de votre serveur. Vèrifiez alors les derniéres lignes de ce fichier :

```
shell> tail host_name.err
shell> tail host_name.log
```

Recherchez des lignes comme celles-ci :

```
000729 14:50:10  bdb:  Recovery function for LSN 1 27595 failed
000729 14:50:10  bdb:  warning: ./test/t1.db: No such file or directory
000729 14:50:10  Can't init databases
```

Cela signifie que vous n'avez pas dèmarrè `mysqld` avec `--bdb-no-recover` et Berkeley DB a trouvè une erreur dans les fichiers de log lorsqu'il a essayè de restaurer votre base. Pour pouvoir continuer, vous devez dèplacer le vieux fichier de log Berkeley DB vers un autre dossier, pour l'examiner plus tard. Les fichiers de logs sont nommès '`log.0000000001`', et ce nombre augmente au fil du temps.

Si vous exècutez `mysqld` avec les tables BDB et que `mysqld` fait des core dumps au dèmarrage, c'est peut être que vous avez des problémes avec le fichier de restauration de BDB. Dans ce cas, essayez de dèmarrer `mysqld` avec `--bdb-no-recover`. Si cela aide, vous devriez alors retirer tous les fichiers de log '`log.*`' du dossier de donnèes, et dèmarrer `mysqld` á nouveau.

Si vous obtenez l'erreur suivant, cela signifie que d'autres programmes (ou un autre serveur `mysqld`) fonctionne dèjá avec le port TCP/IP ou la socket que `mysqld` essaie d'utiliser :

```
Can't start server: Bind on TCP/IP port: Address already in use
  ou
Can't start server : Bind on unix socket...
```

Utilisez `ps` pour vous assurer que vous n'avez pas d'autre serveur `mysqld` qui fonctionne. SI vous ne pouvez pas trouver d'autre serveur en fonctionnement, essayer d'exècuter la commande `telnet votre-nom-d-hote numero-de-port-tcp` puis pressez la touche 'Entrèe' plusieurs fois. SI vous n'obtenez pas de message d'erreur comme `telnet: Unable to connect to remote host: Connection refused`, alors un autre processus utilise le port TCP/IP de `mysqld`. Voyez Section 2.4.1 [mysql_install_db], page 101 et Section 4.1.4 [Multiple servers], page 202.

Si `mysqld` est en fonctionnement, vous pouvez connaître les chemins qu'il utilise avec la commande suivante :

```
shell> mysqladmin variables
```

ou

```
shell> mysqladmin -h 'your-host-name' variables
```

Si vous obtenez une erreur `Errcode 13`, qui indique `Permission denied`, lors du dèmarrage de `mysqld`, cela signifie que vous n'avez pas les droits de lecture ou d'ècriture sur le dossier de donnèes de MySQL, ou dans le dossier de logs. Dans ce cas, vous devriez dèmarrer `mysqld` en tant que root, ou changer les permissions des fichiers et dossiers utilisès.

Si `safe_mysqld` dèmarre le serveur, mais que vous n'arrivez pas á vous y connecter, vous devriez vous assurer que vous avez une entrèe dans le fichier '`/etc/hosts`' qui ressemble á ceci :

```
127.0.0.1       localhost
```

Ce probléme survient uniquement sur les systémes qui n'ont pas une librairie de threads fonctionnels, ou pour lesquels MySQL a ètè configurè pour utiliser les MIT-pthreads.

SI vous ne pouvez pas faire dèmarrer `mysqld`, essayez de faire un fichier de trace pour identifier le probléme. Voir Section E.1.2 [Making trace files], page 816.

Si vous utilisez les tables `InnoDB`, reportez vous aux options de configuration spècifiques á `InnoDB`. Voir Section 7.5.2 [InnoDB start], page 545.

Si vous utilisez les tables BDB (Berkeley DB), vous devriez vous familiariser avec les options de dèmarrage spècifiques á BDB. Voir Section 7.6.3 [BDB start], page 586.

2.4.3 Lancer et arrêter MySQL automatiquement.

Les scripts `mysql.server` et `safe_mysqld` peuvent être utilisès pour dèmarrer le serveur automatiquement au moment du dèmarrage du serveur. `mysql.server` peut aussi servir á arrêter le serveur.

Le script `mysql.server` peut servir á dèmarrer ou arrêter le serveur en l'appelant avec les arguments `start` ou `stop` :

```
shell> mysql.server start
shell> mysql.server stop
```

`mysql.server` est installè dans le dossier '`share/mysql`' du dossier d'installation de MySQL, ou dans le dossier '`support-files`' de la distribution source.

Avant que `mysql.server` ne dèmarre le serveur, il change de dossier pour aller dans le dossier d'installation et appelle `safe_mysqld`. Vous pourriez avoir á èditer `mysql.server` si vous avez une installation binaire dans une situation non standard. Modifiez la commande `cd` avec le dossier correct, avant qu'il n'exècute `safe_mysqld`. Si vous voulez que le serveur fonctionne avec un utilisateur spècifique, ajouter l'option `user` approprièe dans le fichier '`/etc/my.cnf`', tel que prèsentè ultèrieurement dans cette section.

`mysql.server stop` arrête le serveur en lui envoyant un signal. Vous pouvez èteindre le serveur manuellement avec la commande `mysqladmin shutdown`.

Vous pouvez ajouter ces commandes de dèmarrage et d'arrêt aux endroits appropriès dans votre fichier '`/etc/rc*`' lorsque vous dèmarrez MySQL dans les applications de production. Notez que si vous modifiez `mysql.server`, et que vous passez á une nouvelle version de MySQL, votre vesrion modifièe sera ècrasèe, et vous devriez faire une copie de sauvegarde de votre script.

Si votre systéme utilise '`/etc/rc.local`' pour dèmarrer des scripts externes, vous devriez ajouter la ligne suivante :

```
/bin/sh -c 'cd /usr/local/mysql ; ./bin/safe_mysqld --user=mysql &'
```

Vous pouvez aussi ajouter des options á `mysql.server` via le fichier global '`/etc/my.cnf`' file. Un fichier '`/etc/my.cnf`' typique peut ressembler á ceci :

```
[mysqld]
datadir=/usr/local/mysql/var
socket=/var/tmp/mysql.sock
port=3306
user=mysql

[mysql.server]
basedir=/usr/local/mysql
```

Le script `mysql.server` comprend les optiosn suivantes : `datadir`, `basedir` et `pid-file`.

La table suivante montre quels groupes d'options chaque scripts de dèmarrage utilise :

Script	**Groupe d'options**
`mysqld`	`mysqld` and `server`
`mysql.server`	`mysql.server`, `mysqld`, and `server`
`safe_mysqld`	`mysql.server`, `mysqld`, and `server`

Voir Section 4.1.2 [Option files], page 198.

2.5 Changer de version de MySQL

Vous pouvez toujours les fichiers de structures et de donnèes entre les diffèrentes versions de MySQL. La version de base actuelle est la version 3. Si vous changez le jeu de caractéres lors de l'utilisation de MySQL (ce qui va aussi affecter le tri), vous devez exècuter la commande `myisamchk -r -q --set-character-set=charset` sur toutes les tables. Sinon, vos index ne seront pas correctement triès.

Si vous avez peur des nouvelles versions, vous pouvez toujours renommer votre vieux `mysqld` avec un nom comme `mysqld-ancienne_version`. Si votre nouveau serveur `mysqld` se comportement bizarrement, vous pourrez toujours l'èteindre, et redèmarrer avec votre vieux `mysqld`!

Lorsque vous faites une èvolution de version, vous devriez toujorus faire une sauvegarde de vos anciennes donnèes.

Si aprés un changement de version, vous rencontrez des problémes avec les clients recompilès, comme `Commands out of sync` ou des core dumps inopinès, vous avez probablement utiliser un vieux fichier d'entête ou une vieille librairie lors de la compilation de vos programmes. Dans ce cas, vèrifiez la date de votre fichier '`mysql.h`', et de votre librairie '`libmysqlclient.a`', pour vous assurer qu'ils proviennent bien de la nouvelle distribution MySQL. Si ce n'est pas le cas, recompilez vos programmes!

Si vous avez des problémes comme si le nouveau serveur `mysqld` qui ne veut pas dèmarrer, ou que vous ne pouvez pas vous connecter sans un mot de passe, vèrifiez que vous n'avez pas un vieux fichier '`my.cnf`' dans votre installation! Vous pouvez le vèrifier comme ceci : `program-name --print-defaults`. Si cette commande affiche autre chose que le nom du programme, vous avez un fichier '`my.cnf`' actif, qui perturbe vos opèrations.

C'est une bonne idèe que de reconstruire et reinstaller le module `Msql-Mysql` á chaque fois que vous faites une nouvelle version de MySQL, en particulier si vous rencontrez des symptômes tels que les `DBI` qui font des core dump aprés votre mise á jour de MySQL.

2.5.1 Passer de la version 3.23 á la version 4.0

En gènèral, ce que vous devez faire pour passer en version 4.0, á partir d'une version plus ancienne :

- Exècutez le script `mysql_fix_privilege_tables` pour ajouter de nouveaux droits et fonctionnalitès á a table MySQL.
- Editez les scripts de dèmarrage MySQL pour les fichiers de configuration pour ne plus utiliser les options abandonnèes, listèes ci-dessous.
- Convertissez vos vieilles tables `ISAM` en tables `MyISAM` avec la commande : `mysql_convert_table_format database`. Notez que cela ne doit être fait que si toutes les tables de la base sont des tables `ISAM` ou `MyISAM`. Si ce n'est pas le cas, vous devrez alors utiliser la commande `ALTER TABLE table_name TYPE=MyISAM` sur toutes les tables `ISAM`.
- Assurez vous que vous n'avez pas de client MySQL qui utilise des librairies partagèes (comme les modules Perl Msql-Mysql). Si vous en avez, vous devriez les recompiler car les structures utilisèes dans '`libmysqlclient.so`' ont changèes.

MySQL 4.0 va fonctionner même si vous ne suivez pas les instructions ci-dessus, mais il ne sera pas capable de profiter des nouveaux droits disponibles avec MySQL 4.0 et vous pourriez rencontrer des problémes lors de l'èvolution vers MySQL 4.1 ou plus rècent. Les fichiers `ISAM` fonctionne toujours en MySQL 4.0 mais il est abandonnè, et il sera dèsactivè en MySQL 5.0.

Les anciens clients doivent fonctionner avec le serveur version 4.0 sans aucun probléme.

Même si vous suivez les instructions ci-dessus, vous pourrez retourner en version MySQL 3.23.52 ou plus rècent, si vous rencontrez des difficultès avec MySQL 4.0. Dans ce cas, vous devez utiliser la commande `mysqldump` sur toutes les tables qui utilisent un index en texte plein, et restaurer ces tables en 3.23 (car la version 4.0 utilise un nouveau format pour les index en texte plein).

Voici une liste plus compléte de points á contrôler lorsque vous passez á la version 4.0 :

- MySQL 4.0 a de trés nombreux nouveaux droits dans la table `mysql.user`. Voir Section 4.3.1 [GRANT], page 226.

  Pour faire fonctionner ces nouveaux droits, vous devez exècuter le script `mysql_fix_privilege_tables`. Jusqu'á ce que ce script soit exècutè, les utilisateurs auront les droits de `SHOW DATABASES`, `CREATE TEMPORARY TABLES`, et `LOCK TABLES`. Les droits de `SUPER` et `EXECUTE` hèritent leur valeur du droit de `PROCESS`. `REPLICATION SLAVE` et `REPLICATION CLIENT` hèritent leur valeur de `FILE`.

  Si vous avez un script qui crèe automatiquement des nouveaux utilisateur, vous devez le modifier pour y inclure les nouveaux droits. Si vous n'utilisez pas la commande `GRANT` dans ces scripts, c'est une bonne idèe que de les vèrifier.

  En version 4.0.2, l'option `--safe-show-database` est abandonnèe (et ne fait plus rien du tout). Voir Section 4.2.3 [Privileges options], page 208.

  Si vous obtenez des interdictions d'accés pour les nouveaux utilisateurs en version 4.0.2, vous devriez vèrifier si vous avez besoin de nouveaux droits que vous n'utilisiez pas avant. En particulier, vous aurez besoin du droit de `REPLICATION SLAVE` (au lieu de `FILE`) pour les nouveaux esclaves.

- Les paramétres de dèmarrage `myisam_max_extra_sort_file_size` et `myisam_max_extra_sort_file_size` sont dèsormais exprimès en octets, et non plus en Mo, comme cela ètait le cas jusqu'en version 4.0.3). Les systémes externes de verrouillages des tables MyISAM/ISAM sont dèsormais dèsactivès par dèfaut. Vous pouvez les rèactiver avec l'option `--external-locking` Pour la plupart des utilisateurs, ce n'est jamais nècessaire.

- Les options de dèmarrage suivantes ont ètè renommèes :

Ancien nom	**Nouveau nom**.
`myisam_bulk_insert_tree_size`	`bulk_insert_buffer_size`
`query_cache_startup_type`	`query_cache_type`
`record_buffer`	`read_buffer_size`
`record_rnd_buffer`	`read_rnd_buffer_size`
`sort_buffer`	`sort_buffer_size`
`warnings`	`log-warnings`

  Les options de dèmarrage `record_buffer`, `sort_buffer` et `warnings` vont encore fonctionner avec MySQL 4.0 mais elles sont obsolétes.

- Les variables SQL suivantes ont changè de nom.

Ancien nom	**Nouveau nom**.
`SQL_BIG_TABLES`	`BIG_TABLES`
`SQL_LOW_PRIORITY_UPDATES`	`LOW_PRIORITY_UPDATES`
`SQL_MAX_JOIN_SIZE`	`MAX_JOIN_SIZE`
`SQL_QUERY_CACHE_TYPE`	`QUERY_CACHE_TYPE`

Les anciens noms fonctionneront encore en MySQL 4.0, mais sont obsolétes.

- Vous devez utiliser la commande `SET GLOBAL SQL_SLAVE_SKIP_COUNTER=#` au lieu de `SET SQL_SLAVE_SKIP_COUNTER=#`.
- Renommez les options de dèmarrage `--skip-locking` en `--skip-external-locking` et `--enable-locking` en `--external-locking`.
- `SHOW MASTER STATUS` retourne dèsormais une liste vide si les logs binaires ne sont pas activès.
- `SHOW SLAVE STATUS` retourne dèsormais une liste vide si l'esclave n'est pas initialisè.
- `mysqld` dispose dèsormais de l'option `--temp-pool` activèe par dèfaut, car cela donne de meilleurs performances sur certains systémes d'exploitation.
- Les colonnes `DOUBLE` et `FLOAT` acceptent dèsormais l'option `UNSIGNED` pour le stockage (auparavant, `UNSIGNED` ètait ignorè pour ces colonnes).
- `ORDER BY column DESC` trie dèsormais les valeurs `NULL` en premier. En version 3.23, ce n'ètait pas toujours le cas.
- `SHOW INDEX` a 2 colonnes de plus (`Null` et `Index_type`) qu'il n'avait pas en version 3.23.
- `SIGNED` est un mot rèservè.
- Le rèsultat de toutes les opèrations sur les bits, `|`, `&`, `<<`, `>>` et `~` est maintenant non signè. Cela peut poser des problémes si vous aviez un contexte dans lequel vous souhaitez un rèsultat signè. Voir Section 6.3.5 [Cast Functions], page 470.
- **Note** : lorsque vous utilisez la soustraction entre des entiers dont l'un est `UNSIGNED`, le rèsultat sera non signè! En d'autres termes, avant de passer á la version MySQL 4.0, vous devriez vèrifier les situations oú votre application soustrait une valeur d'un entier non signè, et que vous attendez une valeur nègative, ou si vous soustrayez une valeur non signèe d'une colonne. Vous pouvez dèsactiver ce comportement en utilisant l'option de dèmarrage `--sql-mode=NO_UNSIGNED_SUBTRACTION` lorsque vous dèmarrez `mysqld`. Voir Section 6.3.5 [Cast Functions], page 470.
- Pour utiliser `MATCH ... AGAINST (... IN BOOLEAN MODE)` avec vos table,s vous devez les reconstruire avec `ALTER TABLE table_name TYPE=MyISAM`, **même** si la table est dèjá au format `MyISAM`.
- `LOCATE()` et `INSTR()` sont sensibles á la casse, si l'un des arguments est une chaîne binaire. Sinon, ils sont insensibles á la casse.
- `STRCMP()` utilise dèsormais le jeu de caractéres courant pour les comparaisons, ce qui signifie que le comportement par dèfaut des comparaisons est dèsormais insensible á la casse.
- `HEX(string)` retourne dèsormais les caractéres convertis sous la forme d'une chaîne hexadècimale. Si vous voulez convertir un nombre en hexadècimal, vous devez vous assurer d'appeler `HEX()` avec un argument numèrique.
- En version 3.23, `INSERT INTO ... SELECT` fonctionne toujours avec l'option `IGNORE`. En version 4.0.1, MySQL va s'arrêter (et peut être annuler la transaction) si vous ne spècifiez pas l'option `IGNORE`.
- '`safe_mysqld`' est renommèe en '`mysqld_safe`'. Pour encore un peu de temps, nous allons inclure dans nos distributions binaires le script `safe_mysqld` vous la forme d'un lien symbolique vers `mysqld_safe`.

- Les fonctions de l'ancienne API C API `mysql_drop_db`, `mysql_create_db` et `mysql_connect` ne sont plus supportèes, á moins que vous ne compiliez MySQL avec `CFLAGS=-DUSE_OLD_FUNCTIONS`. Au lieu de cela, il sera plus sage de changer vos programmes, pour qu'il utilisent la nouvelle API 4.0.
- Dans la structure `MYSQL_FIELD`, `length` et `max_length` ont èvoluè de `unsigned int` en `unsigned long`. Cela ne va pas causer de problémes, hormis le fait qu'ils peuvent gènèrer des messages d'alerte lorsqu'ils sont utilisè comme argument de fonctions comme `printf()`.
- Vous devriez utiliser la commande `TRUNCATE TABLE` lorsque vous voulez effacer toutes les lignes d'une table, et que vous ne souhaitez pas savoir combien de lignes ont ètè effacèes de la table (car `TRUNCATE TABLE` est plus rapide que `DELETE FROM table_name`).
- Vous allez rencontrer une erreur si vous avez un verrou actif ou une transaction active, et que vous essayez d'utiliser les commandes `TRUNCATE TABLE` ou `DROP DATABASE`.
- Vous devriez utiliser des entiers pour stocker les valeurs dans les colonnes de type BIGINT (au lieu d'utiliser des chaînes, comme vous le faisiez en MySQL 3.23). Utiliser des chaînes va toujours fonctionner, mais passer des entiers est bien plus efficace.
- Le format de `SHOW OPEN TABLE` a ètè changè.
- Les clients multi-threadès doivent utiliser `mysql_thread_init()` et `mysql_thread_end()`. Voir Section 8.4.8 [Threaded clients], page 660.
- Si vous voulez recompiler le module Perl DBD::mysql, vous devez prendre les versions Msql-Mysql-modules 1.2218 ou plus rècente, car les anciennes versions des module DBD utilisent une fonction `drop_db()` abandonnèe.
- `RAND(seed)` retourne un nombre diffèrent en version 4.0 qu'en version 3.23 : cela est fait pour diffèrencier plus fortement `RAND(seed)` de `RAND(seed+1)`.

2.5.2 Passer de la version 3.22 á la version 3.23

MySQL version 3.23 supporte les nouvelles tables `MyISAM` et les anciennes tables `ISAM`. Vous n'avez pas á convertir vos anciennes tables pour utiliser la nouvelle version 3.23. Par dèfaut, toutes les nouvelles tables seront crèèes avec le type `MyISAM` (á moins que vous ne lanciez `mysqld` avec l'option `--default-table-type=isam`). Vous pouvez changer la table `ISAM` en table `MyISAM` avec la commande `ALTER TABLE table_name TYPE=MyISAM` ou le script Perl `mysql_convert_table_format`.

Les clients des versions 3.22 et 3.21 vont fonctionner sans problémes avec la verison 3.23 du serveur.

La liste suivante indique les points á vèrifier lors de la migration :

- Toutes les tables qui utilisent le jeu de caractéres `tis620` doivent être corrigèes avec `myisamchk -r` ou `REPAIR TABLE`.
- Si vous exècutez une commande `DROP DATABASE` sur un lien symbolique, le lien et la base originale seront effacès. Cela n'arrivait pas en 3.22 car `configure` ne dètectait pas les appels á `readlink`.
- `OPTIMIZE TABLE` ne fonctionne que pour les tables `MyISAM`. Pour les autres types de tables, vous devez utiliser `ALTER TABLE` pour optimiser la table. Durant la commande `OPTIMIZE TABLE`, la table est verrouillèe.

- Le client MySQL `mysql` est dèmarrè par dèfaut avec l'option option `--no-named-commands (-g)`. Cette option peut être dèsactivèe avec `--enable-named-commands (-G)`. Cela peut causer des problémes d'incompatibilitè dans certains cas : par exemple, dans les scripts SQL qui utilisent des commandes nommèes sans point virgule! Le format long de la commande devrait fonctionner correctement.
- Les fonctions de date qui travaille sur des parties de dates (comme `MONTH()`) vont dèsormais retourner 0 pour la date `0000-00-00`. (MySQL 3.22 retournait `NULL`.)
- Si vous utilisez le jeu de caractéres `allemand` pour les tris, vous devez rèparer vos tables avec `isamchk -r`, car nous avons fait des modifications dans l'ordre de tri.
- Le type de retour par dèfaut de `IF` dèpendant maintenant des deux arguments, et plus seulement du premier.
- `AUTO_INCREMENT` ne fonctionne pas sur les nombres nègatifs. La raison pour cela est que les nombres nègatifs posaient des problémes d'ècrasement entre -1 et 0. `AUTO_INCREMENT` pour les tables MyISAM est maintenant gèrè á un niveau plus bas, et il est bien plus rapide. Pour les tables MyISAM, les anciens numèros ne sont plus rèutilisès, même si vous effacez des lignes dans la table.
- `CASE`, `DELAYED`, `ELSE`, `END`, `FULLTEXT`, `INNER`, `RIGHT`, `THEN` et `WHEN` sont de nouveaux mots rèservès.
- `FLOAT(X)` est maintenant un vèritable type de nombre á virgule flottante, avec un nombre dèfini de dècimales.
- Lors de la dèclaration de `DECIMAL(length,dec)`, la taille de l'argument n'inclut plus une place pour le signe ou le sèparateur dècimal.
- Une chaîne `TIME` doit être fournie au format suivant : `[[[DAYS] [H]H:]MM:]SS[.fraction]` ou `[[[[[H]H]H]H]MM]SS[.fraction]`.
- `LIKE` compare maintenant les chaînes en appliquant les mêmes régles que `=`. Si vous voulez l'ancien comportement, vous pouvez compiler MySQL avec l'option `CXXFLAGS=-DLIKE_CMP_TOUPPER`.
- `REGEXP` est maintenant insensible á la casse pour les chaînes normales (non binaires).
- Quand vous vèrifiez/rèparez des tables, vous devez utiliser `CHECK TABLE` ou `myisamchk` pour les tables `MyISAM` (‘`.MYI`’) et `isamchk` pour les tables `ISAM` (‘`.ISM`’).
- Si vous voulez que les fichiers d'export de `mysqldump` soit compatibles entre les versions MySQL 3.22 et 3.23, vous ne devez pas utiliser l'option `--opt` ou `--all` de `mysqldump`.
- Vèrifiez tous vos appels á `DATE_FORMAT()` pour vous assurer qu'il y a un signe pourcentage ‘`%`’ avant chaque caractére de format (MySQL version 3.22 et plus rècent avait dèjá cette syntaxe).
- `mysql_fetch_fields_direct` est maintenant une fonction (c'ètait une macro), qui retourne un pointeur sur `MYSQL_FIELD` au lieu de `MYSQL_FIELD`.
- `mysql_num_fields()` ne peut plus être utilisè sur les objets `MYSQL*` (c'est maintenant une fonction qui prend `MYSQL_RES*` comme argument. Il faut donc utiliser `mysql_field_count()` á la place).
- En MySQL version 3.22, le rèsultat de `SELECT DISTINCT ...` ètait toujours triè. En version 3.23, vous devez spècifier la clause `GROUP BY` ou `ORDER BY` pour obtenir un rèsultat triè.

- `SUM()` retourne dèsormais `NULL`, au lieu de 0, si il n'y a pas de lignes á calculer. Ceci s'accorde avec la norme SQL.
- `AND` ou `OR` avec les valeurs `NULL` vont dèsormais retourner `NULL` au lieu de 0. Cela affecte surtout les requêtes qui utilisait `NOT` ou une expression `AND/OR` telle que `NOT NULL = NULL`. `LPAD()` et `RPAD()` vont rèduire la taille de la chaîne rèsultante, si elle est plus grand que l'argument de taille.

2.5.3 Passer de la version 3.21 á la version 3.22

Rien qui n'affecte la compatibilitè n'a changè entre les versions 3.21 et 3.22. Le seul probléme courant est que les nouvelles tables qui sont crèèes avec le type `DATE` vont dèsormais utiliser le nouveau format de stockage. Vous ne pourrez pas accèder á ces nouveaux formats depuis les vieilles versions de `mysqld`.

Aprés avoir installè MySQL version 3.22, vous devriez dèmarrer le nouveau serveur, et exècuter le script `mysql_fix_privilege_tables`. Il va ajouter les nouveaux droits á la commande `GRANT`. SI vous oubliez cela, vous obtiendrez des erreurs `Access denied` lorsque vous essayez d'utiliser les commandes `ALTER TABLE`, `CREATE INDEX` ou `DROP INDEX`. Si votre compte `root` MySQL utilise un mot de passe, vous devriez l'indiquer au script `mysql_fix_privilege_tables`.

L'interface C de `mysql_real_connect()` a changè. Si vous avez un vieux client qui appelle cette fonction, vous devez placer un `0` pour le nouvel argument `db` (ou rèècrire le client pour qu'il envoie l'èlèment `db`, et accèlére les connexions). Vous devez aussi appeler `mysql_init()` avant d'appeler `mysql_real_connect()`! Ce changement a ètè fait pour permettre l'appel de la fonction `mysql_options()`, qui sauve les options dans la structure `MYSQL`.

La variable `key_buffer` de `mysqld` a changè de nom, et est devenue `key_buffer_size`, mais vous pouvez toujours utiliser l'ancien nom dans vos fichiers de dèmarrage.

2.5.4 Passer de la version 3.20 á la version 3.21

Si vous avez une version de MySQL plus ancienne que la version 3.20.28 et que vous voulez passer á la version 3.21, vous devez suivre ces ètapes :

Vous pouvez dèmarrer le serveur `mysqld` version 3.21 avec le script `safe_mysqld --old-protocol` pour l'utiliser avec les clients de la version 3.20. Dans ce cas, la fonction `mysql_errno()` des nouveaux clients ne sera pas fonctionnelle, et seul `CR_UNKNOWN_ERROR` (mais il fonctionne pour les erreurs client), et le serveur utilisera l'ancienne fonction `password()` plutôt que la nouvelle.

Si vous **n'utilisez pas** l'option `--old-protocol` avec `mysqld`, vous devez suivre ces instructions :

- Tous les clients doivent être recompilès. Si vous utilisez ODBC, vous devez obtenir le nouveau pilote `MyODBC` 2.x.
- Le script `scripts/add_long_password` doit être utilisè pour convertir le champs `Password` de la table `mysql.user` en `CHAR(16)`.
- Tous les mots de passe doivent être rèassignès dans la table `mysql.user` pour utiliser les mots de 62 bits au lieu de 31 bits.

- Le format de table n'a pas changè, ce qui vous èvite d'avoir á convertir des tables.

MySQL version 3.20.28 et plus rècent peut gèrer les nouvelles tables `user` sans affecter les clients. Si vous avez une version plus ancienne que la 3.20.28, les mots de passe ne seront plus valide, si vous convertissez la table `user`. Pour être tranquille, commencez par faire passer votre version á la 3.20.28 puis passez en version 3.21.

Le nouveau client fonctionne avec le serveur 3.20.x `mysqld`, alors si vous rencontrez des problémes avec la version 3.21.x, vous pouvez toujours vous rabattre sur les vieux serveurs 3.20.x sans recompiler les clients.

Si vous n'utilisez pas l'option `--old-protocol` de `mysqld`, les vieux clients vont èmettre une erreur :

```
ERROR: Protocol mismatch. Server Version = 10 Client Version = 9
```

La nouvelle interface Perl `DBI/DBD` supporte aussi l'ancienne interface `mysqlperl`. Le seul changement que vous devez faire si vous utilisez `mysqlperl` est de changer les arguments de la fonction `connect()`. Les nouveaux arguments sont : `host`, `database`, `user`, et `password` (les arguments `user` et `password` ont ètè èchangès). Voir Section 8.2.2 [Perl `DBI` Class], page 592.

Les modifications actuelles affectent les requêtes des anciennes applications :

- `HAVING` doit être spècifiè avant la clause `ORDER BY`.
- Les paramétres de la fonction `LOCATE()` ont ètè èchangès.
- Il y a de nouveaux mots rèservès. Les plus notables sont `DATE`, `TIME` et `TIMESTAMP`.

2.5.5 Migrer depuis une autre architecture

Si vous utilisez MySQL version 3.23, vous pouvez copier les fichiers '`.frm`', '`.MYI`' et '`.MYD`' entre les diffèrentes architectures qui supportent le même format de nombre á virgule flottante (MySQL prend en charge les èchanges d'octets).

Les donnèes MySQL des tables `ISAM` et les fichiers d'index ('`.ISD`' et '`*.ISM`', respectivement) sont dèpendantes de l'architecture, et dans certains cas, dèpendantes du systéme d'exploitation. Si vous voulez dèplacer des applications vers une autre machine qui a une autre architecture, ou un autre systéme d'exploitation que votre machine courante, il est recommandè de ne pas faire une simple copie de base en copiant les fichiers vers leur nouvelle destination. Utilisez plutôt `mysqldump`.

Par dèfaut, `mysqldump` va crèer un fichier de requêtes SQL. Vous pouvez alors transfèrer le fichier sur une autre machine, et le fournir comme script á un client `mysql`.

Essayez la commande `mysqldump --help` pour voir quelles options sont disponibles. Si vous envoyez les donnèes vers une nouvelle version de MySQL, il recommandè d'utiliser l'option `mysqldump --opt` pour obtenir un export compact et plus rapide.

Le plus facile (mais pas le plus rapide) pour dèplacer une base de donnèes entre deux machines et d'exècuter les commandes suivantes sur la machine qui hèberge la base :

```
shell> mysqladmin -h 'other hostname' create db_name
shell> mysqldump --opt db_name \
        | mysql -h 'other hostname' db_name
```

Si vous voulez copier la base depuis une machine distante sur un rèseau lent, vous pouvez utiliser :

```
shell> mysqladmin create db_name
shell> mysqldump -h 'other hostname' --opt --compress db_name \
        | mysql db_name
```

Vous pouvez aussi stocker le rèsultat dans un fichier, et transfèrer le fichier sur la machine de destination, puis charger ce fichier dans le serveur. Par exemple, vous pouvez exporter la base vers un fichier source comme ceci :

```
shell> mysqldump --quick db_name | gzip > db_name.contents.gz
```

Le fichier crèè est compressè. Transfèrez le fichier contenant le contenu de votre base sur la machine de destination, puis utilisez ces commandes :

```
shell> mysqladmin create db_name
shell> gunzip < db_name.contents.gz | mysql db_name
```

Vous pouvez aussi utiliser `mysqldump` et `mysqlimport` pour accomplir cette opèration. Pour les grandes tables, c'est bien plus rapide que d'utiliser simplement `mysqldump`. Dans les commandes suivantes, `DUMPDIR` reprèsente le chemin complet du dossier que vous utilisez pour stocker le rèsultat de `mysqldump`.

Premiérement, crèez un dossier pour les fichiers d'exportation, puis faites l'export :

```
shell> mkdir DUMPDIR
shell> mysqldump --tab=DUMPDIR db_name
```

Puis transfèrez les fichiers du dossier `DUMPDIR` dans un dossier correspondant, dans la machine de destination, puis chargez ces fichiers dans MySQL comme ceci :

```
shell> mysqladmin create db_name           # Crèation de la base
shell> cat DUMPDIR/*.sql | mysql db_name   # Crèation des tables dans la base
shell> mysqlimport db_name DUMPDIR/*.txt   # Chargement des donnèes dans les tables
```

N'oubliez pas non plus de copier le contenu de votre base `mysql` car c'est lá que rèsident les droits (`user`, `db`, `host`). Vous devrez alors exècuter les commandes en tant que `root` MySQL sur la nouvelle machine, jusqu'á ce que vous ayez rèinstallè `mysql`.

Aprés l'importation de la base `mysql` sur la nouvelle machine, exècutez la commande `mysqladmin flush-privileges` pour que le serveur relise les droits.

2.6 Notes spècifiques aux systémes d'exploitation

2.6.1 Notes relatives á Linux (toutes versions)

Les notes suivantes relatives á `glibc` ne s'appliquent que dans la situation oú vous construisez MySQL vous-même. Si vous n'utilisez pas Linux sur une machine x86 machine, dans la plupart des cas, il sera mieux pour vous d'utiliser nos binaires. Nous lions nos binaires avec la meilleur version patchèe de `glibc` que nous pouvons fournir et avec les meilleurs options du compilateur, en essayant de les rendre bons pour un serveur qui connait de fortes charges. Et donc, si vous lisez le texte suivant et que vous avez un doute sur ce que vous devez faire, essayez d'abord notre binaire pour voir s'il vous convient, et ne vous souciez de vosz propres constructions qu'aprés vous être assurès que notre binaire ne rèpond pas á vos attentes. Dans ce cas, nous apprècierons une note á propos de cela, pour que nous puissions faire mieux la prochaine fois. Pour les besoins d'un utilisateur de base, pour des

configurations avec beaucoup de connexions et/ou des tables dèpassant la limite des 2G, notre binaire est dans la plupart des cas le meilleur choix.

MySQL utilise LinuxThreads sur Linux. Si vous utilisez une vielle version de Linux qui ne posséde pas `glibc2`, vous devz installer LinuxThreads avant d'essayer de compiler MySQL. Vous pouvez obtenir LinuxThreads á l'adresse suivante : `http://www.mysql.com/Downloads/Linux/`.

Note : Nous avons rencontrè quelques problémes ètranges avec Linux 2.2.14 et MySQL sur les systémes SMP. Si vous avez un systéme SMP, nous vous recommandons de mettre á jour á Linux 2.4 dés que possible ! Votre systéme n'en sera que plus rapide et plus stable !

Notez que les versions de `glibc` infèrieure ou ègale á la 2.1.1 ont un bogue fatal dans la gestion de `pthread_mutex_timedwait`, qui est utilisè lors que vous exècutez `INSERT DELAYED`. Nous vous recommandons de ne pas utiliser `INSERT DELAYED` avant de mettre á jour glibc.

Si vous planifiez d'avoir plus de 1000 connexions simultanèes, vous aurez besoin d'apporter quelques modifications á LinuxThreads, le recompiler, et relier MySQL avec le nouveau '`libpthread.a`'. Augmentez `PTHREAD_THREADS_MAX` dans '`sysdeps/unix/sysv/linux/bits/local_lim.h`' á 4096 et diminuez `STACK_SIZE` dans '`linuxthreads/internals.h`' á 256 KB. Les chemins sont relatifs á la racine de `glibc`. Notez que MySQL ne sera pas stable autour de 600-1000 connexions si `STACK_SIZE` est á 2 MB (par dèfaut).

Si MySQL n'arrive pas á ouvrir assez de fichiers, ou á crèer assez de connexions, il se peut que vous n'ayez pas configurè Linux pour qu'il gére assez de fichiers.

Dans Linux 2.2 ou plus, vous pouvez connaitre le nombre de gestionnaires de fichiers allouès en faisant :

```
cat /proc/sys/fs/file-max
cat /proc/sys/fs/dquot-max
cat /proc/sys/fs/super-max
```

Si vous avez plus de 16 MB de mèmoire, vous devez ajouter qauelque chose comme ce qui suit dans vos scripts d'initialisation ('`/etc/init.d/boot.local`' sur SuSE Linux) :

```
echo 65536 > /proc/sys/fs/file-max
echo 8192 > /proc/sys/fs/dquot-max
echo 1024 > /proc/sys/fs/super-max
```

Vous pouvez aussi exècuter les commandes prècèdentes á partir de la ligne de commande en tant que root, mais les changements seront perdus au prochain redèmarrage de l'ordinateur.

Vous pouvez sinon dèfinir ces paramétres lors du dèmarrage de la machine en utilisant l'outil `sysctl`, qui est utilisè par plusieurs distributions Linux (SuSE l'a aussi ajoutè, á partir de SuSE Linux 8.0). Ajoutez simplement les valeurs suivantes dans un fichier nommè '`/etc/sysctl.conf`' :

```
# Increase some values for MySQL
fs.file-max = 65536
fs.dquot-max = 8192
fs.super-max = 1024
```

Vous devez aussi ajouter ce qui suit á '`/etc/my.cnf`' :

```
[safe_mysqld]
open-files-limit=8192
```

Cela devrait permettre á MySQL de crèer jusqu'á 8192 fichiers de conexion.

La constante `STACK_SIZE` des LinuxThreads contrôle l'espacement des piles de threads dans l'espace d'adressage. Elle doit être assez grande pour qu'il y ait plusieurs chambres pour la pile de chaque thread individuel, mais assez petite pour empêcher les piles de certains threads d'agir sur les donnèes globales de `mysqld`. Malheureusement, l'implèmentation Linux de `mmap()`, comme nous l'avons dècouvert, will successfully unmap an already mapped region if you ask it to map out an address already in use, zeroing out the data on the entire page, instead of returning an error. So, the safety of `mysqld` or any other threaded application depends on the "gentleman" behaviour of the code that creates threads. The user must take measures to make sure the number of running threads at any time is sufficiently low for thread stacks to stay away from the global heap. With `mysqld`, you should enforce this "gentleman" behaviour by setting a reasonable value for the `max_connections` variable.

Si vous construisez MySQL vous-mêmes et ne voulez pas vous amuser á patcher LinuxThreads, vous ne devez pas dèpasser 500 pour la valeur de `max_connections`. Cela devrait même être moins si vous avez un tampon de clefs assez large, de grosses tables heap, ou d'autres choses qui peuvent faire allouer beaucoup de mèmoire á `mysqld`, ou si vous utilisez un noyau 2.2 avec un patch 2G. Si vous utilisez notre binaire ou RPM 3.23.25 ou plus, vous pouvez mettre `max_connections` á 1500 sans problémes, en supposant que vous n'avez ni de grosses tables heap ni grands tampons de clefs. Plus vous rèduirez `STACK_SIZE` dans LinuxThreads plus les threads crèès seront sûrs. Nous recommandons une valeur entre 128K et 256K.

Si vous utilisez beaucoup de connexions simultanèes vous souffrirez peut-être d'une "fonctionnalitè" du noyau 2.2 qui pènalise un processus lors du fork ou du clonage d'un enfant en essayant de prèvenir un attaque du type fork bomb. This will cause MySQL not to scale well as you increase the number of concurrent clients. On single-CPU systems, we have seen this manifested in a very slow thread creation, which means it may take a long time to connect to MySQL (as long as 1 minute), and it may take just as long to shut it down. On multiple-CPU systems, we have observed a gradual drop in query speed as the number of clients increases. In the process of trying to find a solution, we have received a kernel patch from one of our users, who claimed it made a lot of difference for his site. The patch is available at `http://www.mysql.com/Downloads/Patches/linux-fork.patch`. We have now done rather extensive testing of this patch on both development and production systems. It has significantly improved `MySQL` performance without causing any problems and we now recommend it to our users who are still running high-load servers on 2.2 kernels. This issue has been fixed in the 2.4 kernel, so if you are not satisfied with the current performance of your system, rather than patching your 2.2 kernel, it might be easier to just upgrade to 2.4, which will also give you a nice SMP boost in addition to fixing this fairness bug.

We have tested MySQL on the 2.4 kernel on a 2-CPU machine and found MySQL scales **much** better—there was virtually no slowdown on queries throughput all the way up to 1000 clients, and the MySQL scaling factor (computed as the ratio of maximum throughput to the throughput with one client) was 180%. We have observed similar results on a 4-CPU system—virtually no slowdown as the number of clients was increased up to 1000, and 300% scaling factor. So for a high-load SMP server we would definitely recommend the 2.4 kernel

at this point. We have discovered that it is essential to run `mysqld` process with the highest possible priority on the 2.4 kernel to achieve maximum performance. This can be done by adding `renice -20 $$` command to `safe_mysqld`. In our testing on a 4-CPU machine, increasing the priority gave 60% increase in throughput with 400 clients.

Nous essayons aussi actuellement d'obtenir des informations sur le bon fonctionnement de `MySQL` sur le noyaux 2.4 sur les systémes 4-voies and 8-voies. Si vous avez accés á un tel systéme et que vous avez effectuès quelques tests de performances, envoyez-nous un mail á `docs@mysql.com` avec les rèsultats, nous les inclurons dans le manuel.

There is another issue that greatly hurts MySQL performance, especially on SMP systems. The implementation of mutex in LinuxThreads in `glibc-2.1` is very bad for programs with many threads that only hold the mutex for a short time. On an SMP system, ironic as it is, if you link MySQL against unmodified `LinuxThreads`, removing processors from the machine improves MySQL performance in many cases. We have made a patch available for `glibc 2.1.3` to correct this behaviour (`http://www.mysql.com/Downloads/Linux/linuxthreads-2.1-patch`).

With `glibc-2.2.2` MySQL version 3.23.36 will use the adaptive mutex, which is much better than even the patched one in `glibc-2.1.3`. Be warned, however, that under some conditions, the current mutex code in `glibc-2.2.2` overspins, which hurts MySQL performance. The chance of this condition can be reduced by renicing `mysqld` process to the highest priority. We have also been able to correct the overspin behaviour with a patch, available at `http://www.mysql.com/Downloads/Linux/linuxthreads-2.2.2.patch`. It combines the correction of overspin, maximum number of threads, and stack spacing all in one. You will need to apply it in the `linuxthreads` directory with `patch -p0 </tmp/linuxthreads-2.2.2.patch`. We hope it will be included in some form in to the future releases of `glibc-2.2`. In any case, if you link against `glibc-2.2.2` you still need to correct `STACK_SIZE` and `PTHREAD_THREADS_MAX`. We hope that the defaults will be corrected to some more acceptable values for high-load MySQL setup in the future, so that your own build can be reduced to `./configure; make; make install`.

We recommend that you use the above patches to build a special static version of `libpthread.a` and use it only for statically linking against `MySQL`. We know that the patches are safe for `MySQL` and significantly improve its performance, but we cannot say anything about other applications. If you link other applications against the patched version of the library, or build a patched shared version and install it on your system, you are doing it at your own risk with regard to other applications that depend on `LinuxThreads`.

Si vous rencontrez des problémes ètranges lors de l'installation de MySQL, ou quoi que ce soit qui s'en rapproche, il est fort possible que cela soit un probléme de librairie ou de compilateur. Si c'est le cas, l'utilisation de notre binaire les rèsoudra.

Un probléme connu avec la distribution binaire est que avec les anciens systémes Linux qui utilisent `libc` (comme RedHat 4.x ou Slackware), vous obtiendrez quelques erreurs non-fatales de rèsolutions de noms d'hôtes. Voir Section 2.6.1.1 [Binary notes-Linux], page 117.

Lors de l'utilisation des LinuxThreads vous verrez un minimum de trois processus en cours. Il s'agit en fait de threads, il y'a a un pour le gestionnaire des LinuxThreads, un pour gèrer les connexions, et un autre pour gèrer les alarmes et les signaux.

Notez que le noyau Linux et la librairie LinuxThread ne peuvent avoir par dèfaut que 1024 threads. Cela signifie que vous ne pouvez avoir que 1021 connexions á MySQL sur un systéme non-patchè. La page `http://www.volano.com/linuxnotes.html` contient des informations pour contourner cette limite.

Si vous voyez un processus de dèmon `mysqld` mort avec `ps`, cela signifie le plus souvent que vous avez trouvè un bogue dans MySQL ou que vous avez une table corrompue. Voir Section A.4.1 [Crashing], page 697.

To get a core dump on Linux if `mysqld` dies with a `SIGSEGV` signal, you can start `mysqld` with the `--core-file` option. Note that you also probably need to raise the `core file size` by adding `ulimit -c 1000000` to `safe_mysqld` or starting `safe_mysqld` with `--core-file-size=1000000`. Voir Section 4.7.2 [`safe_mysqld`], page 292.

Si vous liez votre propre client MySQL et que vousobtenez l'erreur suivante :

```
ld.so.1: ./my: fatal: libmysqlclient.so.4:
open failed: No such file or directory
```

lors de son exècution, le probléme peut être contournè des façons suivantes :

- Liez le client avec les options suivantes (au lieu de `-Lpath`) : `-Wl,r/path-libmysqlclient.so`.
- Copiez `libmysqclient.so` dans '`/usr/lib`'.
- Ajoutez le chemin vers le rèpertoire oú se trouve '`libmysqlclient.so`' á la variable d'environnement `LD_RUN_PATH` avant de mettre en marche votre client.

Si vous utilisez le compilateur Fujitsu `(fcc / FCC)` vous aurez quelques problémes en compilant MySQL car les fichiers d'entêtes Linux sont trés orientès `gcc`.

La ligne suivante de `configure` devrait fonctionner avec `fcc/FCC` :

```
CC=fcc CFLAGS="-O -K fast -K lib -K omitfp -Kpreex -D_GNU_SOURCE \
-DCONST=const -DNO_STRTOLL_PROTO" CXX=FCC CXXFLAGS="-O -K fast -K lib \
-K omitfp -K preex --no_exceptions --no_rtti -D_GNU_SOURCE -DCONST=const \
-Dalloca=__builtin_alloca -DNO_STRTOLL_PROTO \
'-D_EXTERN_INLINE=static __inline'" ./configure --prefix=/usr/local/mysql \
--enable-assembler --with-mysqld-ldflags=-all-static --disable-shared \
--with-low-memory
```

2.6.1.1 Notes relatives á Linux pour les distributions binaires

MySQL requiére au moins la version 2.0 de Linux.

Attention : Certains utilisateurs de MySQL nous ont avertis qu'ils ont rencontrè de graves problémes de stabilitè avec MySQL et le noyau 2.2.14 de Linux. Si vous utilisez ce noyau, vous devez mettre á jour á la 2.2.19 (ou plus rècent) ou a un noyau 2.4. Si vous utilisez un ordinateur multi-processeurs, vous devriez sèrieusement songer á passer au noyau 2.4 qui vous aportera de grandes performances niveau vitesse.

La version binaire est liée avec `-static`, ce qui signifie que normalement vous n'avez pas besoin de vous soucier des versions des librairies systéme que vous avez. Vous n'avez pas besoin d'installer LinuxThreads non plus. Un programme liè avec `-static` est lègérement plus grand qu'un programme liée dynamiquement mais aussi un peu plus rapide (3-5%).

Un probléme, toutefois, est que vous ne pouvez utiliser de fonctions dèfinies par l'utilisateur avec un programme liè statiquement. Si vous allez ècrire ou utiliser des fonctions UDF (c'est rèservè aux dèveloppeurs C ou C++), vous devez compiler MySQL vous-mêmes, en utilisant les liaisons dynamiques.

Si vous utilisez un systéme basè sur `libc` (au lieu de `glibc2`), vous aurez probablement quelques problémes de rèsolution des noms d'hôtes et des problémes avec `getpwnam()` avec les versions binaires. (Cela vient du fait que `glibc` dèpend malheureusement de quelques librairies externes pour rèsoudre les noms d'hôtes et `getpwent()`, même quand elle est compilèe avec `-static`). Dans ce cas, vous obtiendrez probablement l'erreur suivante quand vous exècuterez `mysql_install_db` :

```
Sorry, the host 'xxxx' could not be looked up
```

ou l'erreur suivante quand vous essayez de dèmarrer `mysqld` avec l'option `--user` :

```
getpwnam: No such file or directory
```

Vous pouvez rèsoudre ce probléme de la façon suivante :

- Obtenez une distribution des sources MySQL (un distribution RPM ou le `tar.gz`) et installez la á la place.
- Exècutez `mysql_install_db --force`; cela n'exècutera pas le test `resolveip` dans `mysql_install_db`. Le mauvais côtè est que vous ne pourrez pas utiliser de noms d'hôtes dans les tables de droits; vous devez utiliser les adresses IP á la place (sauf pour `localhost`). Si vous utilisez une vielle version de MySQL qui ne supporte pas `--force`, vous devez supprimer le test `resolveip` dans `mysql_install` á l'aide d'un èditeur.
- Dèmarrez `mysqld` avec `su` au lieu d'utiliser `--user`.

Le binaire Linux-Intel et les RPM de MySQL sont configurès pour la vitesse la plus grande possible. Nous essayons toujours d'utiliser le compilateur le plus rapide disponible.

Le support Perl de MySQL requiére la version 5.004_03 de Perl ou plus rècent.

Sur quelques version de Linux 2.2, vous pouvez obtenir l'erreur `Resource temporarily unavailable` quand vous faites beaucoup de nouvelles connexions á un serveur `mysqld` en utilisant TCP/IP.

Le probléme est que Linux posséde un dèlai entre votre fermeture de la socket TCP/IP et sa libèration par le systéme. Vu qu'il y'a un nombre fini de places pour les branchements TCP/IP, vous obtiendrez l'erreur prècèdente si vous essayez de faire beaucoup de connexions TCP/IP en peu de temps, comme quand vous exècutez le benchmark MySQL '`test-connect`' via TCP/IP.

Nous avons envoyè des questions plusieurs fois á propos de ce probléme á diffèrentes listes de diffusions Linux mais n'avons jamais rèussi á rèsoudre ce probléme proprement.

Le seul 'correctif' connu pour ce probléme est d'utiliser des connexions persistantes dans vos clients ou d'utiliser les sockets, si vous utilisez le serveur de bases de donnèes et le client sur la même machine. Nous espèrons que le noyau de `Linux 2.4` corrigera ce probléme bientôt.

2.6.1.2 Notes relatives á Linux x86

MySQL requiére la version 5.4.12 de `libc` ou plus rècent. Il est connu pour fonctionner avec `libc` 5.4.46. La version 2.0.6 de `glibc` ou plus rècente devrait aussi fonctionner. Il

y'a eu quelques problémes avec les RPM de `glibc` de RedHat, et donc, si vous avez des problémes, vèrifiez s'il existe des mises á jour ! Les RPM de `glibc` 2.0.7-19 et 2.0.7-29 snot connus pour fonctionner.

Si vous utilisez gcc 3.0 ou plus rècent pour compiler MySQL, vous devez installer la librairie `libstdc++v3` avant de compiler MySQL; si vous ne le faites pas vous obtiendrez une erreur á propos d'un symbole `__cxa_pure_virtual` manquant durant la liaison !

Sur quelques vieilles distributions de Linux, `configure` peut produire une erreur comme celle qui suit :

```
Syntax error in sched.h. Change _P to __P in the /usr/include/sched.h file.
See the Installation chapter in the Reference Manual.
```

Faites ce que le message d'erreur dit et ajoutez un _ á la macro `_P` qui n'en a qu'un, puis essayez á nouveau.

Vous pouvez obtenir quelques avertissements en compilant; celles qui suivent peuvent être ignorèes :

```
mysqld.cc -o objs-thread/mysqld.o
mysqld.cc: In function 'void init_signals()':
mysqld.cc:315: warning: assignment of negative value '-1' to
'long unsigned int'
mysqld.cc: In function 'void * signal_hand(void *)':
mysqld.cc:346: warning: assignment of negative value '-1' to
'long unsigned int'
```

Sur Debian GNU/Linux, si vous voulez que MySQL dèmarre automatiquement lors du dèmarrage de votre systéme, faites ce qui suit :

```
shell> cp support-files/mysql.server /etc/init.d/mysql.server
shell> /usr/sbin/update-rc.d mysql.server defaults 99
```

`mysql.server` peut être trouvè dans le dossier '`share/mysql`' dans le dossier d'installation de MySQL ou dans le dossier '`support-files`' de l'arborescence des sources de MySQL.

Si `mysqld` provoque toujours un core dump au dèmarrage, le probléme peut être que vous avez un vieux '`/lib/libc.a`'. Renommez le, puis supprimez '`sql/mysqld`' et faites á nouveau un `make install` puis rèassayez. Ce probléme a ètè reportè sur quelques installations de Slackware.

Si vous obtenez l'erreur suivante en liant `mysqld`, cela signifie que votre '`libg++.a`' n'est pas installè correctement :

```
/usr/lib/libc.a(putc.o): In function '_IO_putc':
putc.o(.text+0x0): multiple definition of '_IO_putc'
```

Vous pouvez èviter d'utiliser '`libg++.a`' en exècutant `configure` comme suit :

```
shell> CXX=gcc ./configure
```

2.6.1.3 Notes relatives á Linux SPARC

Sur quelques implèmentations, `readdir_r()` est cassè. Le symptome est que `SHOW DATABASES` retourne toujours un rèsultat vide. Cela peut être corrigè en supprimant `HAVE_READDIR_R` de '`config.h`' aprés avoir configurè et avant de commencer á compiler.

Quelques problémes demanderont le patchage de votre installation Linux. Le patch peut être trouvè sur `http://www.mysql.com/Downloads/patches/Linux-sparc-2.0.30.diff`. Ce patch est pour la distribution '`sparclinux-2.0.30.tar.gz`' de Linux qui est disponible sur `vger.rutgers.edu` (une version de Linux qui n'a jamais ètè mèlangèe avec la 2.0.30 officielle). Vous devez aussi installer la version 0.6 des LinuxThreads ou une version plus rècente.

2.6.1.4 Notes relatives á Linux Alpha

La version 3.23.12 de MySQL est la premiére version de MySQL á être testèe sur Linux-Alpha. Si vous voulez utiliser MySQL sur Linux-Alpha, vous devez vous assurer d'avoir cette version ou une version plus rècente.

Nous avons testè MySQL sur Alpha avec nos benchmarks et notre suite de tests, et cela semble fonctionner correctement.

Nous construisons actuellement les packages binaires de MySQL sur SuSE Linux 7.0 pour AXP, kernel 2.4.4-SMP, Compaq C compiler (V6.2-505) et Compaq C++ compiler (V6.3-006) sur une machine Compaq DS20 avec un processeur Alpha EV6.

Vous pouvez trouver les prècèdents compilateurs sur `http://www.support.compaq.com/alpha-tools/`). En utilisant ces compilateurs, au lieu de gcc, nous obtenons á peu prés 9-14% de meilleures performances avec MySQL.

Notez que jusqu'au version 3.23.52 et 4.0.2 de MySQL nous avons optimisè le binaire pour le CPU courant seulement (en utilisant l'option de compilation `-fast`); cela signifiait que vous ne pouviez utiliser nos binaires si vous n'aviez pas un processeur Alpha EV6.

Avec les realeases suivantes nous avons ajoutè l'option `-arch generic` á nos options de compilation, ce qui assure que le binaire fonctionne sur tout les processeurs Alpha. Nous compilons aussi statiquement pour èviter les problémes de librairies.

```
CC=ccc CFLAGS="-fast -arch generic" CXX=cxx \
CXXFLAGS="-fast -arch generic -noexceptions -nortti" \
./configure --prefix=/usr/local/mysql --disable-shared \
--with-extra-charsets=complex --enable-thread-safe-client \
--with-mysqld-ldflags=-non_shared --with-client-ldflags=-non_shared
```

Si vous voulez utiliser egcs la ligne de configuration suivante a fonctionnè pour nous :

```
CFLAGS="-O3 -fomit-frame-pointer" CXX=gcc \
CXXFLAGS="-O3 -fomit-frame-pointer -felide-constructors \
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql \
--disable-shared
```

Quelques problémes connus lors de l'utilisation de MySQL sur Linux-Alpha:

- Le dèbogage d'applications thrèadèes comme MySQL ne fonctionnera pas avec `gdb 4.18`. Vous devez tèlècharger et utiliser gdb 5.1 á la place !
- Si vous essayez de lier statiquement `mysqld` en utilisant `gcc`, l'image rèsultante videra son noyau (core dump) au dèmarrage. En d'autres termes, **n'**utilisez **pas** `--with-mysqld-ldflags=-all-static` avec `gcc`.

2.6.1.5 Note relative á Linux PowerPC

MySQL devrait fonctionner sur MkLinux avec le dernier package `glibc` (testè avec `glibc` 2.0.7).

2.6.1.6 Notes relatives á Linux MIPS

Pour faire fonctionner MySQL sur Qube2, (Linux Mips), vous aurez besoin de la librairie `glibc` la plus rècente (`glibc-2.0.7-29C2` est connue pour marcher). Vous devez aussi utiliser le compilateur `egcs` C++ (`egcs-1.0.2-9`, `gcc 2.95.2` ou plus rècent).

2.6.1.7 Notes relatives á Linux IA64

Pour pouvoir compiler MySQL sous Linux IA64, nous utilisons les lignes de compilation suivante : En utilisant `gcc-2.96` :

```
CC=gcc CFLAGS="-O3 -fno-omit-frame-pointer" CXX=gcc \
CXXFLAGS="-O3 -fno-omit-frame-pointer -felide-constructors \
-fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql \
"--with-comment=Official MySQL binary" --with-extra-charsets=complex
```

Sur IA64 les binaires des clients MySQL utilisent des librairies partagèes. Cela signifie que si vous installez notre distribution binaire á un autre endroit que '`/usr/local/mysql`' vous devez modifier le fichier '`/etc/ld.so.conf`' ou ajouter le chemin vers le rèpertoire oú vous avez '`libmysqlclient.so`' á la variable d'environnement `LD_LIBRARY_PATH`.

Voir Section A.3.1 [Link errors], page 695.

2.6.2 Notes relatives á Windows

Cette section dècrit l'utilisation de MySQL sur Windows. Ces informations sont aussi fournies dans le fichier '`README`' disponible avec la distribution Windows de MySQL.

2.6.2.1 Dèmarrer MySQL sous Windows 95, 98 ou Me

MySQL utilise TCP/IP pour connecter le client au serveur (cela permet á une machine distante de se connecter á votre serveur MySQL). A cause de cela, vous devez installer TCP/IP sur votre machine avant de dèmarrer MySQL. Vous pouvez trouver TCP/IP sur votre CDRom Windows.

Notez que si vous utilisez une vieille version de Windows 95 (par exemple, OSR2), il est probable que vous ayez une vieille version du Winsock; MySQL requiert Winsock 2! Vous pouvez tèlècharger le dernier Winsock sur `http://www.microsoft.com/`. Windows 98 dispose de la nouvelle librairie Winsock 2, et ce paragraphe ne s'applique pas.

Pour dèmarrer le serveur `mysqld`, vous devez utiliser une fenêtre MS-DOS et y taper :

```
C:\> C:\mysql\bin\mysqld
```

Cela va dèmarrer `mysqld` en tâche de fond, sans fenêtre.

Vous pouvez arrêter MySQL avec la commande :

```
C:\> C:\mysql\bin\mysqladmin -u root shutdown
```

Ce script appelle l'utilitaire d'administration MySQL sous le nom de 'root', qui est l'administrateur par dèfaut. Notez bien que les noms d'utilisateurs de MySQL sont totalement indèpendants des utilisateurs Windows.

Notez que Windows 95/98/Me ne supporte pas la crèation des pipes nommès. Sur ces plateformes, vous ne pouvez utiliser que des pipes pour vous connecter á un serveur MySQL distant, qui tournerait sur un hôte Windows NT/2000/XP (le serveur MySQL soit aussi supporter les pipes nommès, bien sur. Par exemple, utiliser `mysqld-opt` sous NT/2000/XP ne permettra pas d'utiliser les pipes nommès. Vous devez utiliser soit `mysqld-nt`, soit `mysqld-max-nt`).

Si `mysqld` ne dèmarre pas, vèrifiez le fichier '`\mysql\data\mysql.err`' pour voir si le serveur n'a pas placè un message d'erreur indiquant la cause du probléme. Vous pouvez aussi essayer de dèmarrer le serveur avec la commande `mysqld --standalone`; dans ce cas, vous pourriez voir apparaître des informations sur l'ècran qui vous aideront á rèsoudre le probléme.

La derniére option est de dèmarrer `mysqld` avec l'option `--standalone --debug`. Dans ce cas, `mysqld` va ècrire le fichier de log '`C:\mysqld.trace`' qui devrait contenir l'explication du probléme. Voir Section E.1.2 [Making trace files], page 816.

Utilisez `mysqld --help` pour afficher toutes les options que `mysqld` comprend!

2.6.2.2 Dèmarrer MySQL sur Windows NT, 2000 ou XP

Pour faire fonctionner MySQL avec TCP/IP sous Windows NT 4, vous devez installer le service pack 3 (ou plus rècent)!

Normalement, vous devez installer MySQL comme un service Windows NT/2000/XP. Dans le cas oú le serveur fonctionnerait dèjá, commencez par le stopper, en utilisant cette commande :

```
C:\mysql\bin> mysqladmin -u root shutdown
```

Cela appelle l'utilitaire d'administation MySQL, avec l'utilisateur '`root`', qui est par dèfaut un `Administrateur` de MySQL. Notez bien que le systéme de droits de MySQL est totalement indèpendant du systéme d'utilisateur de Windows.

Maintenant, l'installation du serveur en tant que service :

```
C:\mysql\bin> mysqld-max-nt --install
```

Si des options sont nècessaires, vous devez les spècifier comme "`Start parameters`" dans l'utilitaire Windows `Services` avant de dèmarrer le service MySQL.

L'utilitaire `Services` (`Windows Service Control Manager`) est disponible dans le `Windows Control Panel` (sous les `Outils d'administration` de Windows 2000). Il est recommandè de fermer les utilitaires de services durant l'installation `--install` ou la suppression `--remove`, pour èviter des erreurs ètranges.

Pour des informations sur quel exècutable serveur utiliser, voyez Section 2.1.2.2 [Windows prepare environment], page 71.

Notez bien que depuis MySQL version 3.23.44, vous avez le choix de configurer le serveur en `Manual` au lieu d'automatique (si vous ne voulez pas que le service dèmarre en même temps que le serveur) :

```
C:\mysql\bin> mysqld-max-nt --install-manual
```

Le service est installè avec le nom `MySQL`. Une fois installè, il peut être immèdiatement dèmarrè depuis l'utilitaire `Services`, ou avec la commande en ligne `NET START MySQL`.

Une fois qu'il fonctionne, `mysqld-max-nt` peut être arrètè avec `mysqladmin`, depuis l'utilitaire de services, ou en utilisant la commande en ligne `NET STOP MySQL`.

Lorsqu'il fonctionne comme service, le systéme d'exploitation va automatiquement arrèter MySQL au moment de l'exticntion. Avec MySQL versions infèrieures á 3.23.47, Windows attendait simplement quelques secondes l'extinction, puis si cela durait trop, il tuait le processus, ce qui causait parfois des problémes. Par exemple, au dèmarrage suivant, le gestionnaire de table `InnoDB` devait faire une restauration aprés crash. Depuis MySQL version 3.23.48, Windows attend plus longtemps que MySQL s'arrête. Si vous notez que ce temps n'est pas suffisant pour votre installation, il est recommandè de ne pas utiliser MySQL sous forme de service, mais á partir de la ligne de commande, et de l'èteindre manuellement avec `mysqladmin shutdown`.

Il y a ce probléme avec Windows NT (mais pas Windows 2000/XP), qui, par dèfaut, attend 20 secondes l'extinction d'un service, puis tue le processus. Vous pouvez augmenter ce dèlai en ouvrant l'èditeur de base de registre '`\winnt\system32\regedt32.exe`' et en modifiant la valeur de `WaitToKillServiceTimeout` dans '`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control`'. Spècifiez une nouvelle valeur en millisecondes, par exemple 120000 pour que Windows NT attende dèsormais jusqu'á 120 secondes.

Notez que lorsqu'il fonctionne comme un service, `mysqld-max-nt` n'a pas d'accés á la console, et ainsi, aucun mesasge n'est disponible. Les erreurs sont alors stockèes dans le fichier '`c:\mysql\data\mysql.err`'.

Si vous avez des problémes á installer `mysqld-max-nt` comme service, essayer de le dèmarrer en utilisant le chemin complet :

```
C:\> C:\mysql\bin\mysqld-max-nt --install
```

Si cela ne fonctionne pas, vous pouvez faire dèmarrer proprement `mysqld-max-nt` en modifiant le chemin dans la base de registre.

Si vous ne voulez pas dèmarrer `mysqld-max-nt` comme service, vous pouvez le dèmarrer comme ceci :

```
C:\> C:\mysql\bin\mysqld-max-nt --standalone
```

ou

```
C:\> C:\mysql\bin\mysqld --standalone --debug
```

Cette derniére mèthode vous donnera une trace de dèbogage dans le fichier '`C:\mysqld.trace`'. Voir Section E.1.2 [Making trace files], page 816.

2.6.2.3 Faire fonctionner MySQL sous Windows

MySQL supporte TCP/IP sur toutes les plate-formes Windows et les tunnels nommès sur NT/2000/XP. Par dèfaut, les tunnels nommès sont utilisès sur NT/2000/XP pour les connexions locales et TCP/IP pour tout les autres cas si le client á TCP/IP d'installè. Le nom d'hôte spècifie le protocole á utiliser :

Nom d'hôte	Protocole
NULL	Sur NT/2000/XP, essayer les tunnels nommès avant;
(aucun)	si cela ne fonctionne pas, utiliser TCP/IP. Sur 9x/Me, TCP/IP est utilisè.
.	Tunnels nommès
localhost	TCP/IP vers l'hôte courant
hostname	TCP/IP

Vous pouvez forcer un client MySQL á utiliser les tunnels nommès en spècifiant l'option `--pipe` ou en spècifiant le nom d'hôte `.` en tant que nom d'hôte. Utilisez l'option `--socket` pour spècifier le nom du tunnel.

Notez qu'á partir de la version 3.23.50, les tunnels nommès ne sont activès que si mysqld est dèmarrè avec `--enable-named-pipe`. Il en est ainsi car certains utilisateurs ont rencontrè des problémes en coupant le serveur MySQL lorsque celui-ci utilise des tunnels nommès.

Vous pouvez vèrifier si MySQL fonctionne en exècutant les commandes suivantes :

```
C:\> C:\mysql\bin\mysqlshow
C:\> C:\mysql\bin\mysqlshow -u root mysql
C:\> C:\mysql\bin\mysqladmin version status proc
C:\> C:\mysql\bin\mysql test
```

Si `mysqld` est lent á rèpondre aux connexions sur Windows 9x/Me, il y'a probablement un probléme avec vos DNS. Dans ce cas, dèmarrez `mysqld` avec `--skip-name-resolve` et utilisez `localhost` et les adresses IP dans les tables de droits MySQL. Vous pouvez aussi èviter les DNS lors de la connexion á un serveur `mysqld-nt` MySQL tournant sur NT/2000/XP en utilisant l'argument `--pipe` pour spècifier l'utilisation des tunnels nommès. Cela fonctionne pour la plupart des clients MySQL.

Il y'a deux versions de l'outil en ligne de commande MySQL :

Binaire	Description
`mysql`	Compilè nativement sur Windows, ce qui fournit des possiblitès trés limitèes d'èdition de texte.
`mysqlc`	Compilè avec le compilateur Cygnus GNU et librairies, ce qui fournit l'èdition `readline`.

Si vous voulez utiliser `mysqlc.exe`, vous devez copier '`C:\mysql\lib\cygwinb19.dll`' dans le rèpertoire systéme de Windows ('`\windows\system`' ou un endroit similaire).

Les priviléges par dèfaut sur Windows donnent un accés á tous les utilisateurs locaux pour toutes les bases de donnèes sans spècifier de mot de passe. Pour rendre MySQL un peu plus sècurisè, vous devez dèfinir un mot de passe pour tout les utilisateurs et supprimer la ligne dans la table `mysql.user` qui contient `Host='localhost'` et `User=''`.

Vous devez aussi ajouter un mot de passe pour le nouvel utilisateur `root`. L'example suivant commence par supprimer l'utilisateur anonyme qui posséde tous les priviléges, puis configure un mot de passe pour l'utilisateur `root` :

```
C:\> C:\mysql\bin\mysql mysql
mysql> DELETE FROM user WHERE Host='localhost' AND User='';
mysql> QUIT
C:\> C:\mysql\bin\mysqladmin reload
C:\> C:\mysql\bin\mysqladmin -u root password votre_mot_de_passe
```

Aprés avoir changè le mot de passe, si vous voulez couper le serveur `mysqld`, vous pouvez le faire en utilisant la commande :

```
C:\> mysqladmin --user=root --password=votre_mot_de_passe shutdown
```

Si vous utilisez la vielle version 3.21 de MySQL sous Windows, la commande prècèdente èchouera avec l'erreur : `parse error near 'SET password'`. La solution est de mettre á jour á la derniére version de MySQL qui est maintenant disponible.

Avec les versions courantes de MySQL vous pouvez facilement ajouter des utilisateurs et changer les priviléges avec les commandes `GRANT` et `REVOKE`. Voir Section 4.3.1 [GRANT], page 226.

2.6.2.4 Connexion á un serveur MySQL distants, sous Windows avec SSH

Voici une note pour connecter un serveur MySQL avec une connexion sècurisèe grâce á SSH (de David Carlson `dcarlson@mplcomm.com`) :

- Installez un client SSH pour votre machine Windows. En tant qu'utilisateur, le meilleur que je connaisse est celui de `SecureCRT` de `http://www.vandyke.com/`. Une autre option est `f-secure` de `http://www.f-secure.com/`. Vous pouvez aussi en trouver d'autres de gratuit avec `Google` á `http://directory.google.com/Top/Computers/Security/Produc and_Tools/Cryptography/SSH/Clients/Windows/`.
- Dèmarrez votre client SSH Windows. Spècifiez `Host_Name = yourmysqlserver_URL_or_IP`. Spècifiez `userid=your_userid` pour vous loger dans votre serveur (probablement avec un mot de passe et un nom d'utilisateur diffèrent).
- Configurez le forward de port. Faites soit un forward distant (spècifiez `local_port: 3306`, `remote_host: yourmysqlservername_or_ip`, `remote_port: 3306`) soit un forward local (spècifiez `port: 3306`, `host: localhost`, `remote port: 3306`).
- Sauvez le tout, sinon vous devrez le refaire la prochaine fois.
- Connectez vous á votre serveur avec la session SSH que vous venez de crèer.
- Sous votre machine Windows, dèmarrez une application ODBC (comme Access).
- Crèez un nouveau fichier dans Windows et reliez le avec MySQL en utilisant le pilote ODBC de la même façon que vous le feriez habituellement, hormis le fait que vous devrez taper `localhost` comme hôte serveur au lieu de `yourmysqlservername`.

Vous avez maintenant une connexion ODBC avec un serveur MySQL distant, et sècurisèe avec SSH.

2.6.2.5 Paratger les donnèes entre plusieurs disque sous Windows

A partir de la version 3.23.16 de MySQL, les serveurs `mysqld-max` et `mysql-max-nt` dans les distributions MySQL sont compilès avec l'option `-DUSE_SYMDIR`. Cela vous permet de placer une base de donnèes sur un disque diffèrent en lui ajoutant un lien symbolique (c'est d'une certaine maniére similaire au fonctionnement des liens symboliques sous Unix).

Sous Windows, vous crèez un lien symbolique vers la base de donnèes en crèant un fichier qui contient le cheminvers le rèpertoire de destination et en le sauvant dans le rèpertoire

'`mysql_data`' sous le nom '`database.sym`'. Notez que le lien symbolique ne sera utilisè que si le rèpertoire '`mysql_data_dir\database`' n'existe pas.

Par exemple, si le rèpertoire des donnèes de MySQL est '`C:\mysql\data`' et que vous voulez avoir une base de donnèes `foo` situèe dans '`D:\data\foo`', vous devez crèer le fichier '`C:\mysql\data\foo.sym`' qui contient le texte `D:\data\foo\`. Aprés cela, toutes les tables crèes dans `foo` le seront dans '`D:\data\foo`'.

Notez qu'á cause de la pènalitè sur la vitesse que vous obtenez en ouvrant toutes les tables, nous n'avons pas activè cela par dèfaut, même si vous avez compilè MySQL avec le support des liens symboliques. Pour les activer, vous devez placer dans votre fichier '`my.cnf`' ou '`my.ini`' l'entrèe suivante :

```
[mysqld]
use-symbolic-links
```

En MySQL 4.0 nous activerons les liens symboliques par dèfaut. Vous devrez alors utiliser l'option `skip-symlink` pour les dèsactiver.

2.6.2.6 Compiler les clients MySQL sous Windows

Dans vos fichiers sources, vous devez inclure '`windows.h`' avant '`mysql.h`' :

```
#if defined(_WIN32) || defined(_WIN64)
#include <windows.h>
#endif
#include <mysql.h>
```

Vous pouvez soit lier votre code avec la librairie dynamique '`libmysql.lib`', qui est juste une interface pour charger '`libmysql.dll`' á la demande, soit lier avec la librairie statique '`mysqlclient.lib`'.

Notez que puisque les librairies mysqlclient sont compilèes en tant que librairies threadèes, vous devez aussi compiler votre code pour qu'il soit multi-thrèadè !

2.6.2.7 MySQL pour Windows face á MySQL pour Unix

MySQL pour Windows a prouvè qu'il ètait trés stable. Cette version de MySQL a les mêmes fonctionnalitès que la version Unix, a quelques exceptions :

Windows 95 et les threads

Windows 95 perf environs 200 octets de mèmoire central lors de la crèation de chaque thread. Chaque connexion MySQL crèe un nouveau thread, ce qui fait qu'il n'est pas recommandè d'exècuter `mysqld` pour des durèes longues sur Windows 95 si votre serveur gére de nombreuses connexions. Les autres versions de Windows ne souffrent pas du même probléme.

Lectures concurrentes

MySQL dèpends des fonctions `pread()` et `pwrite()` pour être capable de mêler des `INSERT` et des `SELECT`. Actuellement, nous utilise mutexes pour èmuler les fonctions `pread()`/`pwrite()`. Nous allons, á long terme, remplacer ce niveau d'interface par une interface virtuelle de façon á ce que nous puissions utiliser l'interface `readfile()`/`writefile()` de Windows NT/2000/XP pour gagner de

la vitesse. L'implèmentation courante limite le nombre de fichiers ouverts par MySQL á 1024, ce qui signifie que vous ne pouvez pas utiliser d'aussi nombreux threads concurrents sur NT/2000/XP que sur Unix.

Blocking read
MySQL utilise une lecture bloquèe pour chaque connexion. Cela signifie que :

- Une connexion ne sera pas dèconnectèe automatiquement aprés 8 heures d'inactivitè, comme c'est le cas sous Unix.
- Si une connexion se bloque, il est impossible de la dètruire sans tuer MySQL.
- `mysqladmin kill` ne fonctionne pas sur une connexion endormie.
- `mysqladmin shutdown` ne peut pas s'exècuter tant qu'il y a des connexions qui dorment.

Nous envisageons de corriger ce probléme, lorsque les dèveloppeurs Windows auront fourni un palliatif.

`DROP DATABASE`
Vous ne pouvez pas dètruire une base qui est utilisèe par un autre thread.

Killing MySQL from the task manager
Vous ne pouvez pas tuer MySQL depuis le gestionnaire de tâche ou avec un utilitaire d'extinction de Windows 95. Vous devez l'èteindre avec `mysqladmin shutdown`.

Noms sensibles á la casse
Les noms de fichiers sont insensibles á la casse sous Windows, ce qui fait que les noms de tables et de bases ne sont pas sensibles á la casse pour MySQL sur Windows. La seule restriction est que les noms de tables et de bases doivent être donnès avec même casse dans le nom (tout en majuscules, ou en minuscules). Voir Section 6.1.3 [Name case sensitivity], page 408.

Le caractére '\'
Les composants d'un chemin sont sèparès par le caractére '\' sous Windows, qui est aussi le caractére de protection de MySQL. Si vous utilisez la commande `LOAD DATA INFILE` ou `SELECT ... INTO OUTFILE`, vous devez doubler le caractére '\' :

```
mysql> LOAD DATA INFILE "C:\\tmp\\skr.txt" INTO TABLE skr;
mysql> SELECT * INTO OUTFILE 'C:\\tmp\\skr.txt' FROM skr;
```

Alternativement, vous pouvez utiliser les noms de fichiers au format Unix, avec le caractére '/' :

```
mysql> LOAD DATA INFILE "C:/tmp/skr.txt" INTO TABLE skr;
mysql> SELECT * INTO OUTFILE 'C:/tmp/skr.txt' FROM skr;
```

`Can't open named pipe` erreur
Si vous utilisez MySQL 3.22 sous NT avec les derniers clients mysql, vous allez obtenir cette erreur :

```
error 2017: can't open named pipe to host: . pipe...
```

Ceci est dû au fait que les versions modernes de MySQL utilisent des pipes nommès sous NT, par dèfaut. Pour èviter cette erreur, vous devez utiliser l'option `--host=localhost` sur les nouveaux clients, ou bien crèer le fichier d'options '`C:\my.cnf`', qui contiendra les informations suivantes :

```
[client]
host = localhost
```

Depuis la version 3.23.50, les pipes nommès sont les seuls activès si `mysqld` est dèmarrè avec `--enable-named-pipe`.

Erreur `Access denied for user`

Si vous rencontrez l'erreur `Access denied for user: 'utilisateur@unknown' to database 'mysql'` lors de l'accés au serveur MySQL sur la même machine, cela signifie que MySQL ne peut rèsoudre proprement votre nom d'hôte.

Pour corriger cela, vous devriez crèer un fichier '`\windows\hosts`' dans l'information suivante :

```
127.0.0.1       localhost
```

`ALTER TABLE`

Lorsque vous exècutez la commande `ALTER TABLE`, la table est verrouillè, empêchant les autres threads d'y accèder. Cela est liè au fait que sous Windows, vous ne pouvez pas effacer un fichier qui est en cours d'utilisation par d'autres threads : á l'avenir, vous pourrions trouver un moyen de contourner ce probléme.

La commande `DROP TABLE` sur une table qui est utilisèe dans le cadre d'un `MERGE` ne fonctionne pas sous Windows, car le gestionnaire de `MERGE` garde la carte des tables cachèe de la couche supèrieure de MySQL. Comme Windows ne vous autorise pas á effacer des fichiers qui sont ouverts, vous devez d'abord vider de la mèmoire toutes les tables du `MERGE` (avec la commande `FLUSH TABLES`) puis effacer la table `MERGE`avant d'effacer les tables. Nous allons corriger cela lorsque nous introduirons la notion de `VIEW`s.

Les directives `DATA DIRECTORY` et `INDEX DIRECTORY` de `CREATE TABLE` sont ignorèes sous Windows, car Windows ne supporte pas les liens symboliques.

Voici quelques problémes connus et pas encore corrigès, si jamais quelqu'un souhaite nous aider sur la version Windows :

- Mettre en place une librairie pour un utilisateur simple `MYSQL.DLL`. Elle inclurait toutes les caractèristiques d'un serveur MySQL standard, hormis la crèation de threads. Cela permettra d'inclure facilement MySQL dans des applications qui n'ont pas vraiment besoin du support client, ou qui n'ont pas besoin d'accèder á un serveur distant.
- Ajouter des icônes sympas pour le dèmarrage et l'arrêt de MysQL, dans l'installateur.
- Lorsque `mysqld` est enregistrè comme service avec `--install` (sous NT) il serait bien de pouvoir ajouter les options par dèfaut de la ligne de commande. Pour le moment, le palliatif consiste á lister les paramétres dans le fichier '`C:\my.cnf`'.
- Il serait vraiment pratique de pouvoir arrêter le processus `mysqld` depuis le gestionnaire de tâches. Pour le moment, il faut passer par `mysqladmin shutdown`.

- Le port de `readline` sur Windows pour pouvoir l'utiliser avec l'outil de ligne de commande `mysql`.
- Des versions graphiques des clients standards MySQL (`mysql`, `mysqlshow`, `mysqladmin` et `mysqldump`) seraient bien.
- Il serait bien si les fonctions de lecture et d'ècriture sur les sockets de '`net.c`' pouvaient être interrompues. Cela rendrait possible l'arrêt des threads en court avec `mysqladmin kill` sous Windows.
- `mysqld` se lance toujours avec la locale "C" et non pas avec la locale par dèfaut. Nous voulons que `mysqld` utilise la configuration de localisation pour l'ordre de tri.
- Ajouter des macros pour utiliser les mèthodes rapides d'incrèment/dècrèment compatibles avec les threads, fourni par Windows.

D'autres spècificitès de Windows sont dècrites dans le fichier '`README`' qui est livrè avec la distribution Windows de MySQL.

2.6.3 Remarques pour Solaris

Sous Solaris, vous pouvez renontrer des problémes avant même d'avoir dèsarchivè la distribution MySQL! Le programme `tar` de Solaris ne peut pas manipuler de noms de fichiers longs, provoquant les messages suivants quand vous dècompressez MySQL :

```
x mysql-3.22.12-beta/bench/Results/ATIS-mysql_odbc-NT_4.0-cmp-db2,\
informix,ms-sql,mysql,oracle,solid,sybase, 0 bytes, 0 tape blocks
tar: directory checksum error
```

Dans ce cas, vous devez utiliser GNU `tar` (`gtar`) pour dèsarchiver la distribution. Vous pouvez en trouver une copie prècompilèe pour Solaris sur `http://www.mysql.com/Downloads/`.

La gestion native des threads Sun fonctionne uniquement depuis Solaris 2.5. Pour les versions 2.4 et antèrieures, MySQL utilisera automatiquement les MIT-pthreads. Voir Section 2.3.6 [MIT-pthreads], page 95.

Vous pouvez rencontrer les erreurs suivantes lors du configure:

```
checking for restartable system calls... configure: error can not run test
programs while cross compiling
```

Cela signifie que l'installation de votre compilateur est dèfectueuse! Dans ce cas, vous devez mettre á jour votre compilateur en faveur d'une version plus rècente. Vous pouvez aussi rèsoudre le probléme en inèrant la ligne suivante dans le fichier '`config.cache`' :

```
ac_cv_sys_restartable_syscalls=${ac_cv_sys_restartable_syscalls='no'}
```

Si vous utilisez Solaris sur une architecture SPARC, nous recommandons `gcc` 2.95.2 comme compilateur. Vous pouvez le trouver sur `http://gcc.gnu.org/`. Notez que `egcs` 1.1.1 et `gcc` 2.8.1 ne fonctionnent pas correctement sur SPARC!

La ligne `configure` recommandèe dans le cas de l'utilisation de `gcc` 2.95.2 est:

```
CC=gcc CFLAGS="-O3" \
CXX=gcc CXXFLAGS="-O3 -felide-constructors -fno-exceptions -fno-rtti" \
./configure --prefix=/usr/local/mysql --with-low-memory --enable-assembler
```

Si vous avez une machine UltraSPARC, vous pouvez gagner 4% de performances supplèmentaires en ajoutant "-mcpu=v8 -Wa,-xarch=v8plusa" á CFLAGS et CXXFLAGS.

Si vous utilisez le compilateur Forte 5.0 (et supèrieur) de Sun, vous pouvez lancer `configure` de la façon suivante :

```
CC=cc CFLAGS="-Xa -fast -native -xstrconst -mt" \
CXX=CC CXXFLAGS="-noex -mt" \
./configure --prefix=/usr/local/mysql --enable-assembler
```

Vous pouvez crèer un binaire 64 bits avec :

```
CC=cc CFLAGS="-Xa -fast -native -xstrconst -mt -xarch=v9" \
CXX=CC CXXFLAGS="-noex -mt -xarch=v9" \
./configure --prefix=/usr/local/mysql --enable-assembler
```

Lors de bancs de tests MySQL, nous avons gagnè 4% en vitesse sur une UltraSPARC en utilisant Forte 5.0 en mode 32 bits plutôt que gcc 3.2 avec les marqueurs -mcpu.

Si vous crèez un binaire 64 bits, il est de 4% plus lent que le binaire 32 bits, mais en contrepartie vous pouvez gèrer davantage de treads et de mèmoire.

Si vous rencontrez des problémes avec `fdatasync` ou `sched_yield`, vous pouvez les rèsoudre en ajoutant `LIBS=-lrt` á la ligne configure.

Le paragraphe suivant ne s'applique qu'aux compilateurs plus anciens que WorkShop 5.3 :

Vous pouvez avoir á modifier le script `configure` et changer la ligne :

```
#if !defined(__STDC__) || __STDC__ != 1
```

en :

```
#if !defined(__STDC__)
```

Si vous activez `__STDC__` avec l'option `-Xc`, le compilateur Sun ne peut pas compiler avec le fichier d'entêtes '`pthread.h`' de Solaris. C'est un bogue de Sun (compilateur ou fichier d'inclusion dèfectueux).

Si `mysqld` gènére les messages d'erreur suivants lorsque vous le lancez, cela est du au fait que vous avez compilè MySQL avec le compilateur de Sun sans activer l'option multi-threads (`-mt`):

```
libc internal error: _rmutex_unlock: rmutex not held
```

Ajoutez `-mt` á `CFLAGS` et `CXXFLAGS` puis rèessayez.

Si vous utilisez la version SFW de gcc (fournie avec Solaris 8), vous devez ajouter '`/opt/sfw/lib`' á la variable d'environnement `LD_LIBRARY_PATH` avant de lancer le configure.

Si vous utilisez le gcc disponible sur `sunfreeware.com`, vous pouvez rencontrer de nombreux problémes. Vous devriez recompiler gcc et les GNU binutils sur la machine á partir de laquel vous les utiliserez, afin d'èviter tout souci.

Si vous obtenez l'erreur suivante lorsque vous compilez MySQL avec `gcc`, cela signifie que votre `gcc` n'est pas configurè pour votre version de Solaris :

```
shell> gcc -O3 -g -O2 -DDBUG_OFF  -o thr_alarm ...
./thr_alarm.c: In function `signal_hand':
./thr_alarm.c:556: too many arguments to function `sigwait'
```

La meilleure chose á faire dans ce cas est d'obtenir la version la plus rècente de `gcc` et de compiler avec votre `gcc` actuel! Au moins pour Solaris 2.5, la plupart des versions binaires de `gcc` ont d'anciens fichiers d'inclusion inutilisables qui planteront les programmes qui utilisent les threads (ainsi probablement d'autres programmes)!

Solaris ne fournit pas de versions statiques de toutes les librairies systéme (`libpthreads` et `libdl`), vous ne pouvez donc pas compiler MySQL avec `--static`. Si vous tentez de le faire, vous obtiendrez l'erreur :

```
ld: fatal: library -ldl: not found
```

ou

```
undefined reference to 'dlopen'
```

ou

```
cannot find -lrt
```

Si de nombreux processus essaient de se connecter trés rapidement á `mysqld`, vous verrez cette erreur dans le journal MySQL :

```
Error in accept: Protocol error
```

Pour èviter cela, vous pouvez lancer le serveur avec l'option `--set-variable back_log=50`. Veuillez noter que `--set-variable` est dèprèciè depuis MySQL 4.0, utilisez uniquement `--back_log=50`. Voir Section 4.1.1 [Command-line options], page 192.

Si vous liez votre propre client MySQL, vous pouvez avoir l'erreur suivante quand vous le lancez :

```
ld.so.1: ./my: fatal: libmysqlclient.so.#:
open failed: No such file or directory
```

Le probléme peut être èvitè avec l'une des mèthodes suivantes :

- Liez le client avec le marqueur suivant (á la place de `-Lpath`) : `-Wl,r/full-path-to-libmysqlclient.so`.
- Copiez '`libmysqclient.so`' dans '`/usr/lib`'.
- Ajoutez le chemin du rèpertoire oú '`libmysqlclient.so`' est installè á la variable d'environnement `LD_RUN_PATH` avant de lancer votre client.

Si vous avez des soucis avec configure qui essaie de lier avec `-lz` et que vous n'avez pas installè `zlib`, vous avez deux solutions :

- Si vous voulez utiliser le protocole compresseè de communication, vous devrez vous procurer et installer zlib sur ftp.gnu.org.
- Configurez avec `--with-named-z-libs=no`.

Si vous utilisez gcc et rencontrez des problémes en chargeant la fonction `UDF` dans MySQL, essayez d'ajouter `-lgcc` á la ligne de liaison de la fonction `UDF`.

Si vous voulez que MySQL se lance automatiquement, vous pouvez copier '`support-files/mysql.server`' dans '`/etc/init.d`' et crèer un lien symbolique pointant dessus et s'appelant '`/etc/rc3.d/S99mysql.server`'.

Comme Solaris ne supporte pas les fichiers core pour les applications `setuid()`, vous ne pouvez pas obtenir un fichier core de `mysqld` si vous utilisez l'option `--user`.

2.6.3.1 Notes relatives á Solaris 2.7/2.8

Vous pouvez normalement utiliser les binaires Solaris 2.6 sur Solaris 2.7 et 2.8. La plupart des fonctionnalitèes de Solaris 2.6 s'appliquent aussi á Solaris 2.7 et 2.8.

Notez que la version 3.23.4 de MySQL et plus doivent être capables de dètecter automatiquement les nouvelles versions de Solaris et d'activer les parades pour rèsoudre les problémes suivants !

Solaris 2.7 / 2.8 ont quelques bogues dans les fichiers inclus. Vous verrez peut-être l'erreur suivante en utilisant `gcc` :

```
/usr/include/widec.h:42: warning: 'getwc' redefined
/usr/include/wchar.h:326: warning: this is the location of the previous
definition
```

Si cela arrive, vous pouvez faire ce qui suit pour rèsoudre ce probléme :

Copiez `/usr/include/widec.h` vers `.../lib/gcc-lib/os/gcc-version/include` et changez la ligne 41 de :

```
#if     !defined(lint) && !defined(__lint)

en

#if     !defined(lint) && !defined(__lint) && !defined(getwc)
```

Alternativement, vous pouvez èditer directement le fichier '`/usr/include/widec.h`'. De toutes façons, aprés avoir apportè la correction, vous devez effacer '`config.cache`' et exècuter `configure` á nouveau !

Si vous obtenez des erreurs comme celles qui suivent quand vous exècutez `make`, c'est parce que `configure` n'a pas dètectè le fichier '`curses.h`' (probablement á cause de l'erreur dans '`/usr/include/widec.h`') :

```
In file included from mysql.cc:50:
/usr/include/term.h:1060: syntax error before ','
/usr/include/term.h:1081: syntax error before ';'
```

La solution est de faire l'une des choses qui suit :

- Configurez avec `CFLAGS=-DHAVE_CURSES_H CXXFLAGS=-DHAVE_CURSES_H ./configure`.
- Editez '`/usr/include/widec.h`' comme indiquè plus haut et rè-exècutez configure.
- Effacez la ligne `#define HAVE_TERM` di fichier '`config.h`' et exècutez `make` á nouveau.

Si vous obtenez une erreur disant que votre programme de liaison ne peut trouver `-lz` lors de la liaison du programme de votre client, le probléme est probablement que votre fichier '`libz.so`' est installè dans '`/usr/local/lib`'. Vous pouvez corriger ceci en utilisant l'une des mèthodes suivantes :

- Ajoutez '`/usr/local/lib`' á `LD_LIBRARY_PATH`.
- Ajoutez un lien vers '`libz.so`' á partir de '`/lib`'.
- Si vous utilisez Solaris 8, vous pouvez installer la zlib optionelle á partir de votre CD Solaris 8.
- Configurez MySQL avec l'option `--with-named-z-libs=no`.

2.6.3.2 Remarques pour Solaris x86

Sous Solaris 2.8 sur x86, `mysqld` va crasher (core dump) si vous l'exècutez 'strip'.

Si vous utilisez `gcc` ou `egcs` sous Solaris x86 et que vous rencontrez des problémes avec des coredumps, lorsqu'il y a de la charge, il est recommandè d'utiliser la commande de `configure` suivante :

```
CC=gcc CFLAGS="-O3 -fomit-frame-pointer -DHAVE_CURSES_H" \
CXX=gcc \
CXXFLAGS="-O3 -fomit-frame-pointer -felide-constructors -fno-exceptions \
-fno-rtti -DHAVE_CURSES_H" \
./configure --prefix=/usr/local/mysql
```

Cela va èviter les problémes avec la librairie `libstdc++` et avec les exceptions C++.

Si cela ne vous aide pas, il est recommandè de compiler une version de dèbogage, et de l'exècuter avec un fichier de trace sous `gdb`. Voir Section E.1.3 [Using gdb on mysqld], page 817.

2.6.4 Notes relatives á BSD

Cette section fournit des informations pour les diffèrentes variètès de BSD, ainsi que les versions spècifiques de celles-ci.

2.6.4.1 Notes relatives á FreeBSD

FreeBSD 3.x est recommandè pour exècuter MySQL vu que le package des threads est plus intègrè.

La façon la plus facile et la plus conseillèe d'installer est d'utiliser les ports du serveur et du client MySQL disponibles sur `http://www.freebsd.org/`.

Les utiliser vous donnera :

- Un MySQL fonctionnant avec toutes les optimisations connues pour votre version active de FreeBSD.
- Configuration et construction automatique.
- Scripts de dèmarrage installès dans /usr/local/etc/rc.d.
- La possibilitè de voir tous les fichiers installès avec pkg_info -L. Et la possibilitè de les effacer tous avec pkg_delete si vous ne voulez plus de MySQL sur cette machine.

Il est recommandè d'utiliser les MIT-pthreads sur FreeBSD 2.x et les threads natifs sur les versions 3 et plus. Il est possible de faire fonctionner le tout avec les threads natifs sur les derniéres versions 2.2.x mais vous rencontrerez probablement des problémes en coupant `mysqld`.

Le '`Makefile`' de MySQL requiére GNU make (`gmake`) pour fonctionner. Si vous voulez compiler MySQL vous devez d'abord installer GNU `make`.

Assurez-vous que votre configuration de la rèsolution des noms est bonne. Sinon, vous airez peut-être quelques problémes lors de la connexion á `mysqld`.

Assurez vous que l'entrèe `localhost` dans le fichier '`/etc/hosts`' est correcte (sinon, vous aurez des problémes pour vous connecter á la base de donnèes). Le fichier '`/etc/hosts`' doit commencer par :

```
127.0.0.1       localhost localhost.votre.domaine
```

La maniére recommandèe de compiler et d'installer MySQL sur FreeBSD avec gcc (2.95.2 et plus) est :

```
CC=gcc CFLAGS="-O2 -fno-strength-reduce" \
CXX=gcc CXXFLAGS="-O2 -fno-rtti -fno-exceptions -felide-constructors \
-fno-strength-reduce" \
./configure --prefix=/usr/local/mysql --enable-assembler
gmake
gmake install
./scripts/mysql_install_db
cd /usr/local/mysql
./bin/mysqld_safe &
```

Si vous vous apercevez que `configure` utilisera les MIT-pthreads, vous devriez lire les notes relatives aux MIT-pthreads. Voir Section 2.3.6 [MIT-pthreads], page 95.

Si vous obtenez une erreur de `make install` disant qu'il ne peut trouver '`/usr/include/pthreads`', `configure` n'a pas dètectè que vous avez besoin des MIT-pthreads. Cela est corrigè en exècutant ces commandes :

```
shell> rm config.cache
shell> ./configure --with-mit-threads
```

FreeBSD est aussi connu pour avoir une petite limite de gestionnaires de fichiers par dèfaut. Voir Section A.2.16 [Not enough file handles], page 694. Dècommentez la section ulimit -n dans safe_mysqld ou enlevez la limite pour l'utilisateur `mysqld` dans /etc/login.conf (et regènèrez le avec cap_mkdb). Assurez-vous aussi de dèfinir la classe approprièe pour cet utilisateur dans le fichier des mots de passe si vous n'utilisez pas celui par dèfaut. (utilisez : chpass nom-utilisateur-mysqld). Voir Section 4.7.2 [`safe_mysqld`], page 292.

Si vous avez beaucoup de mèmoire, vous devriez penser á recompiler le noyau pour permettre á MySQL d'utiliser plus de 512M de RAM. Regardez l'`option MAXDSIZ` dans le fichier de configuration de LINT pour plus d'informations.

Si vous avez des problémes avec la date courante dans MySQL, configurer la variable d'environnement `TZ` aidera sûrement. Voir Annexe F [Environment variables], page 828.

Pour obtenir un systéme sècurisè et stable, vous ne devez utiliser que les noyaux FreeBSD marquès `-RELEASE`.

2.6.4.2 Notes concernant NetBSD

Pour compiler sur NetBSD vous aurez besoin de GNU `make`. Sinon, la compilation stoppera lorsque `make` essayera d'exècuter `lint` sur les fichiers C++.

2.6.4.3 Notes relatives á OpenBSD 2.5

Dans la version 2.5 de OpenBSD, vous pouvez compiler MySQL avec les threads natifs avec les options suivantes :

```
CFLAGS=-pthread CXXFLAGS=-pthread ./configure --with-mit-threads=no
```

2.6.4.4 Notes relatives á OpenBSD 2.8

Nos utilisateurs nous ont informè que OpenBSD 2.8 comporte un bogue des threads qui pose quelques problémes avec MySQL. Les dèveloppeurs d'OpenBSD ont rèsolu ce probléme, mais depuis le 25 janvier 2001 ce n'est disponible que dans la branche "-current". Les symptômes de ce bogue sont : rèponses lentes, beaucoup de charge, grande utilisation du CPU, et crashes.

Si vous obtenez une erreur comme `Error in accept:: Bad file descriptor` ou erreur 9 en essayant d'ouvrir les tables ou les dossiers, le probléme est probablement que vous n'avez pas allouè assez de descripteurs de fichiers á MySQL.

Dans ce cas, essayez de dèmarrer `safe_mysqld` en tant que root avec les options suivantes :

```
--user=mysql --open-files-limit=2048
```

2.6.4.5 Notes relatives aux versions 2.x de BSD/OS

Si vous obtenez l'erreur suivante lors de la compilation de MySQL, votre valeur de `ulimit` pour la mèmoire vortuelle est trop petite :

```
item_func.h: In method `Item_func_ge::Item_func_ge(const Item_func_ge &)':
item_func.h:28: virtual memory exhausted
make[2]: *** [item_func.o] Error 1
```

Essayez d'utiliser `ulimit -v 80000` et exècutez `make` á nouveau. Si cela ne fonctionne pas et que vous utilisez `bash`, essayez de passer á `csh` ou `sh`; quelques utilisateurs de BSDI ont reportè des problémes avec `bash` et `ulimit`.

Si vous utilisez `gcc`, vous aurez peut-être aussi á utiliser l'option `--with-low-memory` de `configure` pour pouvoir compiler '`sql_yacc.cc`'.

Si vous avez des problémes avec la date courante dans MySQL, configurer la variable `TZ` vous aidera probablement. Voir Annexe F [Environment variables], page 828.

2.6.4.6 Notes relatives aux versions 3.x de BSD/OS

Mettez á jour á la version 3.1 de BSD/OS. Si cela n'est pas possible, installez le patch BSDIpatch M300-038.

Utilisez la commande suivante lors de la configuration de MySQL :

```
shell> env CXX=shlicc++ CC=shlicc2 \
       ./configure \
           --prefix=/usr/local/mysql \
           --localstatedir=/var/mysql \
           --without-perl \
           --with-unix-socket-path=/var/mysql/mysql.sock
```

Ce qui suit fonctionne aussi :

```
shell> env CC=gcc CXX=gcc CXXFLAGS=-O3 \
       ./configure \
           --prefix=/usr/local/mysql \
           --with-unix-socket-path=/var/mysql/mysql.sock
```

Vous pouvez changer les rèpertoires si vous voulez, ou utiliser les valeurs par dèfaut en ne spècifiant pas de chemins.

Si vous avez des problémes de performances alors que la charge est petite, essayez d'utiliser l'option `--skip-thread-priority` de `mysqld` ! Cela exècutera tous les threads avec la même prioritè; Sur la version 3.1 de BSDI, cela donne de meilleures performances (en attendant que BSDI corrige sont gestionnaire de threads).

Si vous obtenez l'erreur `virtual memory exhausted` durant la compilation, vous devez essayer en utilisant `ulimit -v 80000` et exècutant `make` á nouveau. Si cela ne fonctionne pas et que vous utilisez `bash`, essayez de passer á `csh` ou `sh`; quelques utilisateurs de BSDI ont reportè des problémes avec `bash` et `ulimit`.

2.6.4.7 Notes relatives aux versions 4.x de BSD/OS

Les versions 4.x de BSDI ont quelques bogues relatifs aux threads. Si vous voulez utiliser MySQL sur ce systéme, vous devez installer tous les patches liès aux threads. vous devez au moins installer M400-023.

Sur quelques systémes avec une version 4.x de BSDI, vous pouvez rencontrer des problémes avec les librairies partagèes. Le symptôme est que vous ne pouvez utiliser aucun programme client, comme par exemple, `mysqladmin`. Dans ce cas, vous devez le reonfigurer pour qu'il n'utilise pas les librairies paratgèes avec l'option `--disable-shared` de configure.

Quelques utilisateurs ont eu avec BSDI 4.0.1 un probléme faisant qu'aprés un bout de temps, le binaire `mysqld` ne peut plus ouvrir de tables. Cela est du au fait qu'un bogue relatif au systéme ou á la librairie fait changer de rèpertoire á `mysqld` sans qu'on ne l'ait demandè !

La solution est soit de mettre á jour vers la version 3.23.34 ou de supprimer la ligne `#define HAVE_REALPATH` de `config.h` aprés avoir exècutè `configure` et avant d'exècuter `make`.

Notez que ce qui prècéde signifie que vous ne pouvez pas crèer de lien symbolique sur un dossier de bases de donnèes vers un autre dossier de bases de donnèes ou lier une table symboliquement vers une autre base de donnèes sur BSDI ! (Crèer un lien symbolique vers un autre disque fonctionne).

2.6.5 Notes relatives á Mac OS X

2.6.5.1 Bêta publique Mac OS X

MySQL devrait fonctionner sans problémes sur la bête publique de Mac OS X (Darwin). Vous n'avez pas besoin du patch pthread pour cet OS !

2.6.5.2 Mac OS X Server

Avant d'essayer d'installer MySQL sur Mac OS X server vous devez d'abord installer le package pthread á partir de `http://www.prnet.de/RegEx/mysql.html`.

Notre binaire pour Mac OS X est compilè sur Rhapsody 5.5 avec la ligne de configuration suivante :

```
CC=gcc CFLAGS="-O2 -fomit-frame-pointer" CXX=gcc CXXFLAGS="-O2 \
-fomit-frame-pointer" ./configure --prefix=/usr/local/mysql \
"--with-comment=Official MySQL binary" --with-extra-charsets=complex \
--disable-shared
```

Vous voudrez peut-être ajouter des alias dans vos fichiers de ressources shell pour accèder á `mysql` et `mysqladmin` á partir de la ligne de commande :

```
alias mysql '/usr/local/mysql/bin/mysql'
alias mysqladmin '/usr/local/mysql/bin/mysqladmin'
```

2.6.6 Notes sur les autres Unix

2.6.6.1 Notes relatives á HP-UX pour les distributions binaires

Quelques distributions binaires de MySQL pour HP-UX sont distribuès en tant que fichier depot HP et archive tar. Pour utiliser le fichier depot, vous devez avoir au moins HP-UX 10.x pour avoir accés aux outils depot de HP.

La version HP de MySQL a ètè compilèe sur un serveur HP 9000/8xx sous HP-UX 10.20, et utilise les MIT-pthreads. Elle est connue pour bien fonctionner sous cette configuration. A partir de la version 3.22.26 de MySQL vous pouvez aussi compiler avec le package de threads natifs HP.

Autres configurations pouvant fonctionner :

- HP 9000/7xx avec HP-UX 10.20+
- HP 9000/8xx avec HP-UX 10.30

Les configurations suivantes semblent ne jamais vouloir fonctionner :

- HP 9000/7xx ou 8xx avec HP-UX 10.x oú x < 2
- HP 9000/7xx ou 8xx avec HP-UX 9.x

Pour installer la distribution, utilisez l'une des commandes suivantes, oú `/chemin/vers/depot` est le chemin complet vers le fichier dèpôt :

- Pour tout installer, serveur, client et outils de dèveloppement :

  ```
  shell> /usr/sbin/swinstall -s /chemin/vers/depot mysql.full
  ```

- Pour n'installer que le serveur :

  ```
  shell> /usr/sbin/swinstall -s /chemin/vers/depot mysql.server
  ```

- Pour n'installer que le package client :

  ```
  shell> /usr/sbin/swinstall -s /chemin/vers/depot mysql.client
  ```

- Pour n'installer que les outils de dèveloppement :

```
shell> /usr/sbin/swinstall -s /chemin/vers/depot mysql.developer
```

Le depot place les binaires et les librairies dans '`/opt/mysql`' et les données dans '`/var/opt/mysql`'.Le depot crèe aussi les entrèes appropriées dans '`/etc/init.d`' et '`/etc/rc2.d`' pour dèmarrer le serveur automatiquement lors du boot. Evidemment, il faut être `root` pour installer.

Pour installer la distribution HP-UX tar.gz, vous devez avoir une copie de GNU `tar`.

2.6.6.2 Notes relatives á la version 10.20 de HP-UX

Il y'a quelques petits problémes que vous pourrez rencontrer lors de la compilation de MySQL sur HP-UX. Nous recommandons l'utilisation de `gcc` au lieu du compilateur natif de HP-UX, car `gcc` produit un meilleur code !

Nous recommandons l'utilisation de gcc 2.95 sur HP-UX. N'utilisez pas les options de haute optimisation (comme -O6) car cela pourrait ne pas être sûr sur HP-UX.

La ligne de configuration suivante devrait fonctionner avec gcc 2.95 :

```
CFLAGS="-I/opt/dce/include -fpic" \
CXXFLAGS="-I/opt/dce/include -felide-constructors -fno-exceptions \
-fno-rtti" CXX=gcc ./configure --with-pthread \
--with-named-thread-libs='-ldce' --prefix=/usr/local/mysql --disable-shared
```

La ligne de configuration suivante devrait fonctionner avec gcc 3.1 :

```
CFLAGS="-DHPUX -I/opt/dce/include -O3 -fPIC" CXX=gcc \
CXXFLAGS="-DHPUX -I/opt/dce/include -felide-constructors -fno-exceptions \
-fno-rtti -O3 -fPIC" ./configure --prefix=/usr/local/mysql \
--with-extra-charsets=complex --enable-thread-safe-client \
--enable-local-infile  --with-pthread \
--with-named-thread-libs=-ldce --with-lib-ccflags=-fPIC
--disable-shared
```

2.6.6.3 HP-UX Version 11.x Notes

Pour les version 11.x de HP-UX nous recommandons MySQL 3.23.15 ou plus rècent.

A cause de quelques bogues critiques dans les librairies standard de HP-UX, vous devez installer les patches suivants avant d'essayer de faire fonctionner MySQL sous HP-UX 11.0 :

```
PHKL_22840 Streams cumulative
PHNE_22397 ARPA cumulative
```

Cela rèsoudra le probléme de l'obtention de `EWOULDBLOCK` á partir de `recv()` et `EBADF` á partir de `accept()` dans les applications threadèes.

Si vous utilisez `gcc` 2.95.1 sur un systéme HP-UX 11.x non-patchè, vous obtiendrez l'erreur :

```
In file included from /usr/include/unistd.h:11,
                 from ../include/global.h:125,
                 from mysql_priv.h:15,
```

```
                 from item.cc:19:
/usr/include/sys/unistd.h:184: declaration of C function ...
/usr/include/sys/pthread.h:440: previous declaration ...
In file included from item.h:306,
                 from mysql_priv.h:158,
                 from item.cc:19:
```

Le probléme est que HP-UX ne dèfinit pas `pthreads_atfork()` avec consistence. Il posséde des prototypes en conflit dans '`/usr/include/sys/unistd.h`':184 et '`/usr/include/sys/pthread.h`':440 (dètails ci-dessous).

Une solution est de copier '`/usr/include/sys/unistd.h`' dans '`mysql/include`' et èditer '`unistd.h`' en le changeant pour qu'il corresponde á la dèfinition dans '`pthread.h`'. Voici les modifications :

```
183,184c183,184
<       extern int pthread_atfork(void (*prepare)(), void (*parent)(),
<                                                 void (*child)());
---
>       extern int pthread_atfork(void (*prepare)(void), void (*parent)(void),
>                                                 void (*child)(void));
```

Aprés cela, la ligne de configuration suivante devrait fonctionner :

```
CFLAGS="-fomit-frame-pointer -O3 -fpic" CXX=gcc \
CXXFLAGS="-felide-constructors -fno-exceptions -fno-rtti -O3" \
./configure --prefix=/usr/local/mysql --disable-shared
```

Si vous utilisez MySQL 4.0.5 avec le compilateur HP-UX, vous pouvez utiliser : (testè avec cc B.11.11.04):

```
CC=cc CXX=aCC CFLAGS=+DD64 CXXFLAGS=+DD64 ./configure --with-extra-character-set=co
```

Vous pouvez ignorer toutes les erreurs de ce type :

```
aCC: warning 901: unknown option: '-3': use +help for online documentation
```

Si vous obtenez l'erreur suivante de `configure` :

```
checking for cc option to accept ANSI C... no
configure: error: MySQL requires a ANSI C compiler (and a C++ compiler).
Try gcc. See the Installation chapter in the Reference Manual.
```

Vèrifiez que le chemin vers le compilateur K&R ne prècéde pas le chemin vers le compilateur C et C++ HP-UX.

Une autre raison qui pourrait vous empêcher de compiler, et le fait de n'avoir pas dèfinit l'option `+DD64` ci-dessus.

2.6.6.4 Notes relatives á IBM-AIX

La dètection automatique de `xlC` est absente de Autoconf, ce qui fait qu'un commande `configure` comme celle qui suit est requise lors de la compilation de MySQL (Cet exemple utilise le compilateur IBM) :

```
export CC="xlc_r -ma -O3 -qstrict -qoptimize=3 -qmaxmem=8192 "
export CXX="xlC_r -ma -O3 -qstrict -qoptimize=3 -qmaxmem=8192"
```

```
export CFLAGS="-I /usr/local/include"
export LDFLAGS="-L /usr/local/lib"
export CPPFLAGS=$CFLAGS
export CXXFLAGS=$CFLAGS

./configure --prefix=/usr/local \
--localstatedir=/var/mysql \
--sysconfdir=/etc/mysql \
--sbindir='/usr/local/bin' \
--libexecdir='/usr/local/bin' \
--enable-thread-safe-client \
--enable-large-files
```

Ce sont les options utilisèes pour compiler la distribution de MySQL qui peut être trouvèe sur `http://www-frec.bull.com/`.

Si vous changez le `-O3` en `-O2` dans la ligne prècèdente, vous devez aussi enlever l'option `-qstrict` (c'est une limitation du compilateur IBM C).

Si vous utilisez `gcc` ou `egcs` pour compiler MySQL, vous **devez** utiliser l'option `-fno-exceptions`, vu que la gestion des exceptions de `gcc`/`egcs` n'est pas sûre pour les threads ! (Cela est testè avec `egcs` 1.1.) Il y'a aussi quelques problémes connus avec l'assembleur d'IBM, qui peuvent lui faire gènèrer du mauvais code lors de son utilisation avec gcc.

Nous recommandons la ligne de `configure` suivante avec `egcs` et `gcc 2.95` sur AIX :

```
CC="gcc -pipe -mcpu=power -Wa,-many" \
CXX="gcc -pipe -mcpu=power -Wa,-many" \
CXXFLAGS="-felide-constructors -fno-exceptions -fno-rtti" \
./configure --prefix=/usr/local/mysql --with-low-memory
```

Le `-Wa,-many` est nècessaire pour que la compilation se passe sans problémes. IBM est au courant de ce probléme mais n'est pas pressè de le corriger á cause de l'existence du palliatif. Nous ne savons pas si `-fno-exceptions` est requise avec `gcc 2.95`, mais comme MySQL n'utilise pas les exeptions et que l'option en question gènére un code plus rapide, nous vous recommandons de toujours utiliser cette option avec `egcs / gcc`.

Si vous obtenez un probléme avec le code de l'assembleur essayez en changeant l'option -mcpu=xxx pour l'adapter á votre processeur. Le plus souvent, on a besoin de power2, power, ou powerpc, et sinon 604 ou 604e. Je ne suis pas positif mais je pense que l'utilisation de "power" sera sûre la plupart du temps, même sur une machine power2.

Si vous ne savez pas quel est votre processeur, exècutez "uname -m", cela vous renverra une chaîne comme "000514676700", avec un format xxyyyyyymmss oú xx et ss sont toujours des zèros, yyyyyy est un identifiant unique du systéme et mm est l'identifiant du CPU Planar. Une liste de ces valeurs peut être trouvèe sur `http://publib.boulder.ibm.com/doc_link/en_US/a_doc_lib/cmds/aixcmds5/uname.htm`. Cela vous donnera un type et un modéle de machine que vous pouvez utiliser pour dèterminer quel type de processeur vous avez.

Si vous avez des problémes avec les signaux (MySQL se termine de maniére imprèvue lors des montèes en charge) vous avez peut-être trouvè un bogue du systéme avec les threads et les signaux. Dans ce cas, vous pouvez demander á MySQL de ne pas utiliser les signaux en configuration avec :

```
shell> CFLAGS=-DDONT_USE_THR_ALARM CXX=gcc \
       CXXFLAGS="-felide-constructors -fno-exceptions -fno-rtti \
       -DDONT_USE_THR_ALARM" \
       ./configure --prefix=/usr/local/mysql --with-debug --with-low-memory
```

Cela n'affecte pas les performances de MySQL, mais comporte un effet secondaire faisant en sorte que vous ne pourrez tuer les clients en ètat "sleeping" sur une connexion avec `mysqladmin kill` ou `mysqladmin shutdown`. A la place, le client se terminera lorsqu'il èmmetra sa prochaine commande.

Sur quelques versions de AIX, lier avec `libbind.a` fait vider son noyau á `getservbyname` (core dump). Il s'agit d'un bogue AIX et doit être remontè á IBM.

Pour AIX 4.2.1 et gcc vous devez apporter les modifications suivantes :

Aprés la configuration, èditez '`config.h`' et '`include/my_config.h`' et changez la ligne qui comporte

```
#define HAVE_SNPRINTF 1
```

en

```
#undef HAVE_SNPRINTF
```

Et finalement, dans '`mysqld.cc`' vous devez ajouter un prototype pour initgoups.

```
#ifdef _AIX41
extern "C" int initgroups(const char *,int);
#endif
```

2.6.6.5 Notes relatives á SunOS 4

Avec SunOS 4, les MIT-pthreads sont requis pour compiler MySQL, ce qui signifie que vous aurez besoin de GNU `make`.

Quelques systémes SunOS 4 ont des problémes avec les librairies dynamiques et `libtool`. Vous pouvez utiliser la ligne suivante de `configure` pour èviter ce probléme :

```
shell> ./configure --disable-shared --with-mysqld-ldflags=-all-static
```

Lors de la compilation de `readline`, vous pouvez obtenir des avertissements á propos de dèfinitions dupliquèes. Vous pouvez les ignorer.

Lors de la compilation de `mysqld`, il y'aura quelques avertissements `implicit declaration of function`. Vous pouvez les ignorer.

2.6.6.6 Notes pour Alpha-DEC-UNIX (Tru64)

Si vous utilisez `egcs` 1.1.2 sur Digital Unix, vous devez passer á `gcc` 2.95.2, car `egcs` connait de sèrieux bugs sur DEC!

Lorsque vous compilez des programmes threadès sous Digital Unix, la documentation recommande l'utilisation de l'option `-pthread` avec `cc` et `cxx` et les librairies `-lmach -lexc` (en plus de `-lpthread`). Vous devriez exècuter le script `configure` comme ceci :

```
CC="cc -pthread" CXX="cxx -pthread -O" \
./configure --with-named-thread-libs="-lpthread -lmach -lexc -lc"
```

Lorsque vous compilez `mysqld`, vous pouvez voir apparaître des alertes comme celles-ci :

```
mysqld.cc: In function void handle_connections()':
mysqld.cc:626: passing long unsigned int *' as argument 3 of
accept(int,sockadddr *, int *)'
```

Vous pouvez les ignorer tranquillement. Elles apparaissent car `configure` ne peut dètecter que des erreurs, et pas des alertes.

Si vous dèmarrez le serveur directement en ligne de commande, vous pouvez rencontrer des problémes d'interruption si vous vous dèconnectez. Lorsque vous vous dèconnectez, les processus en cours recoivent le signal `SIGHUP`. Si c'est le cas, essayez de dèmarrer le serveur comme ceci :

```
shell> nohup mysqld [options] &
```

`nohup` fait que la commande suivante va ignorer les signaux `SIGHUP` envoyès par le terminal. Alternativement, vous pouvez dèmarrer le serveur avec le script `safe_mysqld`, qui appelle le dèmon `mysqld` avec l'option `nohup` pour vous. Voir Section 4.7.2 [`safe_mysqld`], page 292.

SI vous avez des problémes pour compiler `mysys/get_opt.c`, vous pouvez simplement supprimer la ligne `#define _NO_PROTO` au dèbut du fichier!

Si vous utilisez le compilateur `cc` de Compaq, la ligne de configuration suivante devrait fonctionner :

```
CC="cc -pthread"
CFLAGS="-O4 -ansi_alias -ansi_args -fast -inline speed all -arch host"
CXX="cxx -pthread"
CXXFLAGS="-O4 -ansi_alias -ansi_args -fast -inline speed all -arch host \
-noexceptions -nortti"
export CC CFLAGS CXX CXXFLAGS
./configure \
--prefix=/usr/local/mysql \
--with-low-memory \
--enable-large-files \
--enable-shared=yes \
--with-named-thread-libs="-lpthread -lmach -lexc -lc"
gnumake
```

Si vous avez un probléme avec `libtool`, lorsque vous compilez les librairies partagèes, ou lorsque vous linkez `mysql`, vous devriez pouvoir rèsoudre ce probléme avec :

```
cd mysql
/bin/sh ../libtool --mode=link cxx -pthread  -O3 -DDBUG_OFF \
-O4 -ansi_alias -ansi_args -fast -inline speed \
-speculate all \ -arch host  -DUNDEF_HAVE_GETHOSTBYNAME_R \
-o mysql  mysql.o readline.o sql_string.o completion_hash.o \
../readline/libreadline.a -lcurses \
../libmysql/.libs/libmysqlclient.so  -lm
cd ..
gnumake
gnumake install
scripts/mysql_install_db
```

2.6.6.7 Notes pour Alpha-DEC-OSF/1

Si vous avez des problémes de compilation et que le `CC` de DEC et `gcc` sont installès, essayez d'utiliser le script `configure` comme ceci :

```
CC=cc CFLAGS=-O CXX=gcc CXXFLAGS=-O3 \
./configure --prefix=/usr/local/mysql
```

Si vous avez des problémes avec le fichier '`c_asm.h`', vous pouvez crèer un fichier inerte '`c_asm.h`' avec :

```
touch include/c_asm.h
CC=gcc CFLAGS=-I./include \
CXX=gcc CXXFLAGS=-O3 \
./configure --prefix=/usr/local/mysql
```

Notez que les problémes suivants avec le programme `ld` peuvent être corrigès en tèlèchargeant le dernier kit de patch de DEC (Compaq) á : `http://ftp.support.compaq.com/public/unix/`.

Su OSF/1 V4.0D et avec le compilateur "DEC C V5.6-071 on Digital Unix V4.0 (Rev. 878)" le compilateur prèsente un comportement ètrange (undefined `asm` symbols). `/bin/ld` apparait aussi comme incorrect (problémes avec des erreurs `_exit undefined` survenant lors du link de `mysqld`). Sur ce systéme, nous avons rèussi á compiler MySQL avec le script `configure` suivant, aprés avoir remplacè `/bin/ld` par la version de OSF 4.0C:

```
CC=gcc CXX=gcc CXXFLAGS=-O3 ./configure --prefix=/usr/local/mysql
```

Avec le compilateur Digital "C++ V6.1-029", la ligne suivante doit fonctionner :

```
CC=cc -pthread
CFLAGS=-O4 -ansi_alias -ansi_args -fast -inline speed -speculate all \
       -arch host
CXX=cxx -pthread
CXXFLAGS=-O4 -ansi_alias -ansi_args -fast -inline speed -speculate all \
          -arch host -noexceptions -nortti
export CC CFLAGS CXX CXXFLAGS
./configure --prefix=/usr/mysql/mysql --with-mysqld-ldflags=-all-static \
            --disable-shared --with-named-thread-libs="-lmach -lexc -lc"
```

Avec certaines versions de OSF/1, la fonction `alloca()` est boguèe. Corrigez cela en supprimant la ligne du fichier '`config.h`' qui dèfinit `'HAVE_ALLOCA'`.

La fonction `alloca()` a aussi un prototype incorrect dans `/usr/include/alloca.h`. L'alerte en rèsultant peut être ignorèe.

Le script `configure` va utiliser automatiquemt les librairies de threads suivantes : `--with-named-thread-libs="-lpthread -lmach -lexc -lc"`.

En utilisant `gcc`, vous pouvez aussi essayer le script `configure` avec ceci :

```
shell> CFLAGS=-D_PTHREAD_USE_D4 CXX=gcc CXXFLAGS=-O3 ./configure ...
```

Si vous avez des problémes avec les signaux (MySQL s'arrête inopinèments sous forte charge), vous pouvez avoir rencontrè un bogue de l'OS avec les threads, et les signaux. Dans ce cas, vous pouvez indiquer á MySQL de ne pas utiliser les signaux avec la configuration suivante :

```
shell> CFLAGS=-DDONT_USE_THR_ALARM \
       CXXFLAGS=-DDONT_USE_THR_ALARM \
       ./configure ...
```

Cela ne modifie pas les performances de MySQL, mais vous ne pourrez plus terminer les clients qui sont en mode "sleeping" sur une connexion avec la commande `mysqladmin kill` ou `mysqladmin shutdown`. Au lieu de cela, le client sera terimnè lorsqu'il èmettra la prochaine commande.

Avec `gcc` 2.95.2, vous aurez probablement les problémes de compilation suivants :

```
sql_acl.cc:1456: Internal compiler error in 'scan_region', at except.c:2566
Please submit a full bug report.
```

Pour corriger cela, vous devez aller dans le dossier `sql` et faire un "copier coller" de la derniére ligne `gcc`, tout en remplaçant le code `-O3` par le code `-O0` ou ajouter le code `-O0` immèdiatement aprés `gcc` si vous n'avez aucune option `-O` sur votre ligne de compilation). Aprés cela, vous pouvez retourner au niveau de la racine de MySQL, et tenter á nouveau un `make`.

2.6.6.8 Notes relatives á SGI Irix

Si vous utilisez la version 6.5.3 d'Irix ou plus rècente, `mysqld` ne pourra crèer de threads que si vous l'exècutez en tant qu'utilisateur possèdant le privilége `CAP_SCHED_MGT` (comme `root`) ou que vous donnez au serveur `mysqld` ce privilége avec la commande suivante :

```
shell> chcap "CAP_SCHED_MGT+epi" /opt/mysql/libexec/mysqld
```

Vous devrez peut-être supprimer quelques dèfinitions dans '`config.h`' aprés avoir exècutè `configure` et avant de compiler.

Sur quelques implèmentations d'Irix, la fonction `alloca()` ne marche pas. Si le serveur `mysqld` se stoppe sur quelques requêtes `SELECT`, supprimez les lignes de '`config.h`' qui dèfinissent `HAVE_ALLOC` et `HAVE_ALLOCA_H`. Si `mysqladmin create` ne fonctionne pas, supprimez la ligne qui dèfinit `HAVE_READDIR_R` dans '`config.h`'. Vous devrez peut-être supprimer la ligne de `HAVE_TERM_H` aussi.

SGI recommande que vous installiez tous les patches de cette page : `http://support.sgi.com/surfzone/patches/patchset/6.2_indigo.rps.html`

Vous devrez, au moins, installer la derniére version du noyau, de `rld` et de `libc`.

Vous avez besoin de tous les patches POSIX sur cette page, pour le support des pthreads :

`http://support.sgi.com/surfzone/patches/patchset/6.2_posix.rps.html`

Si vous obtenez une erreur se rapprochant de la suivante lors de la compilation de '`mysql.cc`':

```
"/usr/include/curses.h", line 82: error(1084): invalid combination of type
```

Tapez ce qui suit dans le rèpertoire racine de votre source MySQL :

```
shell> extra/replace bool curses_bool < /usr/include/curses.h \
> include/curses.h
shell> make
```

Un probléme de planification a aussi ètè signalè. Si seul un thread est en cours, les choses ralentissent. Evitez cela en dèmarrant un autre client. Cela pourra accèlèrer l'exècution de

l'autre thread de 2 á 10 fois. Ceci est un probléme pas encore trés clair avec les threads Irix; vous devrez improviser pour trouver des solutions en attendant que cela soit corrigè.

Si vous compilez avec `gcc`, vous pouvez utiliser la commande `configure` suivante :

```
CC=gcc CXX=gcc CXXFLAGS=-O3 \
./configure --prefix=/usr/local/mysql --enable-thread-safe-client \
--with-named-thread-libs=-lpthread
```

Sous Irix 6.5.11 avec les compilateurs natifs Irix C et C++ versions 7.3.1.2, ce qui suit est connu pour fonctionner :

```
CC=cc CXX=CC CFLAGS='-O3 -n32 -TARG:platform=IP22 -I/usr/local/include \
-L/usr/local/lib' CXXFLAGS='-O3 -n32 -TARG:platform=IP22 \
-I/usr/local/include -L/usr/local/lib' ./configure \
--prefix=/usr/local/mysql --with-innodb --with-berkeley-db \
--with-libwrap=/usr/local \
--with-named-curses-libs=/usr/local/lib/libncurses.a
```

2.6.6.9 Notes pour Caldera (SCO)

Le port courant a ètè testè avec les systémes "sco3.2v5.0.4" et "sco3.2v5.0.5". Il y a eu aussi beaucoup de progrés sur le port de "sco 3.2v4.2".

Pour le moment, le compilateur recommandè sur OpenServer est `gcc` 2.95.2. Avec lui, vous devriez pouvoir compiler MySQL simplement avec :

```
CC=gcc CXX=gcc ./configure ... (options)
```

1. Pour OpenServer 5.0.X, vous devez utiliser `gcc-2.95.2p1` ou plus rècent, de Skunkware. `http://www.caldera.com/skunkware/` et choisissez le visualiseur de packages OpenServer ou via FTP á `ftp://ftp2.caldera.com/pub/skunkware/osr5/devtools/gcc/`
2. Vous avez besoin du port `GCC` 2.5.x pour ce produit, et du systéme de dèveloppement. Ils sont nècessaires sur cette version de Caldera (SCO) Unix. Vous ne pouvez pas juste utiliser le systéme `gcc` dev.
3. Vous devez obtenir le package FSU Pthreads et l'installer en premier. Il est disponible á `http://www.cs.wustl.edu/~schmidt/ACE_wrappers/FSU-threads.tar.gz`. Vous pouvez aussi obtenir un package prècompilè á `http://www.mysql.com/Downloads/SCO/FSU-threads-3.5c.tar.gz`.
4. FSU Pthreads peut être compilè sur Caldera (SCO) Unix 4.2 avec tcpip. Ou OpenServer 3.0 ou Open Desktop 3.0 (OS 3.0 ODT 3.0), avec le Caldera (SCO) Development System installè, utilisant le bon port de `GCC` 2.5.x ODT ou OS 3.0, vous aurez besoin du bon port de `GCC` 2.5.x Il y a beaucoup de problémes sans un bon port. Le port de ce produit requiert le systéme SCO Unix Development. Sans cela, vous manquerez des librairies et des linkeurs nècessaires.
5. Pour compiler FSU Pthreads sur votre systéme, faites ceci :
   a. Exècutez `./configure` dans le dossier '`threads/src`' et sèlectionnez l'option SCO OpenServer. Cette commande copie '`Makefile.SCO5`' dans '`Makefile`'.
   b. Exècutez `make`.

   c. Pour faire l'installation dans le dossier par dèfaut '`/usr/include`', connectez vous en `root`, puis utilisez `cd` vers le dossier '`thread/src`', et exècutez la commande `make install`.

6. N'oubliez pas d'utiliser GNU `make` pour compiler MySQL.
7. Si vous ne dèmmarez pas `safe_mysqld` en tant que `root`, vous aurez probablement la limite de 110 fichiers ouverts par processus. `mysqld` va ècrire une note á ce propos de le fichier de log.
8. Avec SCO 3.2V5.0.5, vous devez utiliser FSU Pthreads version 3.5c ou plus rècent. Vous devez aussi utiliser `gcc` 2.95.2 ou plus rècent!

   La commande `configure` suivante doit fonctionner :

   ```
   shell> ./configure --prefix=/usr/local/mysql --disable-shared
   ```

9. Avec SCO 3.2V4.2, vous devez utiliser FSU Pthreads version 3.5c ou plus rècent. La commande `configure` suivante doit fonctionner :

   ```
   shell> CFLAGS="-D_XOPEN_XPG4" CXX=gcc CXXFLAGS="-D_XOPEN_XPG4" \
          ./configure \
              --prefix=/usr/local/mysql \
              --with-named-thread-libs="-lgthreads -lsocket -lgen -lgthreads" \
              --with-named-curses-libs="-lcurses"
   ```

   Vous pouvez avoir des problémes avec certains fichiers d'inclusion. Dans ce cas, vous pouvez trouver un nouveau fichier d'inclus spècifique á SCO á `http://www.mysql.com/Downloads/SCO/SCO-3.2v4.2-includes.tar.gz`. Vous devrez dècompresser de fichier dans le dossier '`include`' de votre dossier source MySQL.

Notes de dèveloppement Caldera (SCO):

- MySQL devrait dètecter automatiquement les FSU Pthreads et compiler `mysqld` avec `-lgthreads -lsocket -lgthreads`.
- Les librairies de dèveloppement Caldera (SCO) development sont rè-entrantes avec les FSU Pthreads. Caldera dit que ses librairies de fonctions sont rè-entrantes, et donc, que FSU Pthreads aussi. FSU Pthreads sur OpenServer essaie d'utiliser le schèma SCO pour en faire des librairies rè-entrantes.
- Les FSU Pthreads (tout au moins la version de `http://www.mysql.com/`) est livrè compilè avec GNU `malloc`. Si vous avez des problémes d'utilisation de mèmoire, assurez vous que '`gmalloc.o`' est inclus dans '`libgthreads.a`' et '`libgthreads.so`'.
- Avec les FSU Pthreads, les appels systémes suivant sont compatibles avec pthreads : `read()`, `write()`, `getmsg()`, `connect()`, `accept()`, `select()` et `wait()`.
- Le CSSA-2001-SCO.35.2 (le patch est indiquè comme un patch de sècuritè dans erg711905-dscr_remap (version 2.0.0)) rend boguè les FSU threads et MySQL instable. Vous devez supprimer ce patch si vous voulez exècuter `mysqld` sur un serveur OpenServer 5.0.6.

Si vous voulez installer DBI sur Caldera (SCO), vous devez èditer le fichier '`Makefile`' dans DBI-xxx et chaque sous-dossier.

Notez que les lignes suivantes prèsupposent que `gcc` 2.95.2 ou plus rècent est installè :

```
Ancien :                                    Nouveau :
CC = cc                                  CC = gcc
CCCDLFLAGS = -KPIC -W1,-Bexport          CCCDLFLAGS = -fpic
CCDLFLAGS = -wl,-Bexport                 CCDLFLAGS =

LD = ld                                  LD = gcc -G -fpic
LDDLFLAGS = -G -L/usr/local/lib          LDDLFLAGS = -L/usr/local/lib
LDFLAGS = -belf -L/usr/local/lib         LDFLAGS = -L/usr/local/lib

LD = ld                                  LD = gcc -G -fpic
OPTIMISE = -Od                           OPTIMISE = -O1

Ancien :
CCCFLAGS = -belf -dy -w0 -U M_XENIX -DPERL_SCO5 -I/usr/local/include

Nouveau :
CCFLAGS = -U M_XENIX -DPERL_SCO5 -I/usr/local/include
```

Ceci est dû au fait que le `Perl dynaloader` ne va pas charger les modules `DBI` si ils ont ètè compilès avec `icc` ou `cc`.

Perl fonctionne au mieux avec `cc`.

2.6.6.10 Notes relatives á la version 7.0 fr Caldera (SCO) Unixware

Vous devez utiliser une version de MySQL supèrieure ou ègale á la 3.22.13 car cette version corrige quelques problémes de port vers Unixware.

Nous avons rèussi á compiler MySQL avec la ligne suivante de `configure` sur la version 7.0.1 de Unixware :

```
CC=cc CXX=CC ./configure --prefix=/usr/local/mysql
```

Si vous voulez utiliser `gcc`, vous devez utiliser la version 2.95.2 de `gcc` ou plus rècente.

Caldera fournit libsocket.so.2 disponible sur `ftp://stage.caldera.com/pub/security/tools` pour des corrections de sècuritè pre-OSR506. Le correctif de telnetd est aussi disponible sur `ftp://stage.caldera.com/pub/security/openserver/CSSA-2001-SCO.10/` comme libsocket.so.2 et libresolv.so.1 avec des instructions pour installer sur les systémes pre-OSR506.

Il est bon d'installer les patches prècèdents avant d'essayer de compiler/utiliser MySQL.

2.6.7 Notes relatives á OS/2

MySQL utilise un certain nombre de fichiers ouverts. A cause de cela, vous devez ajouter un ligne se rapprochant de la suivante dans votre fichier '`CONFIG.SYS`' :

```
SET EMXOPT=-c -n -h1024
```

Si vous ne le faites pas, vous obtiendrez probablement l'erreur :

```
File 'xxxx' not found (Errcode: 24)
```

Lors de l'utilisation de MySQL avec OS/2 Warp 3, FixPack 29 ou plus est requis. Avec OS/2 Warp 4, FixPack 4 ou plus est requis. C'est un besoin de la librairie des Pthreads. MySQL doit être installè sur une partition qui supporte les noms de fichiers longs, tel que HPFS, FAT32, etc.

Le script '`INSTALL.CMD`' doit être exècutè á partir du '`CMD.EXE`' d'OS/2 et ne fonctionnera probablement pas avec des substitus tel que '`4OS2.EXE`'.

Le script '`scripts/mysql-install-db`' a ètè renommè. Il est maintenant nommè '`install.cmd`' et est un script REXX, qui mettra en place les configurations de sècuritè par dèfaut de MySQL et crèera les icônes WorkPlace Shell pour MySQL.

Le support des module dynamiques est compilè, mais n'est pas assez testè. Les modules dynamiques doivent être compilès en utilisant la librairie run-time Pthreads.

```
gcc -Zdll -Zmt -Zcrtdll=pthrdrtl -I../include -I../regex -I.. \
    -o exemple udf_exemple.cc -L../lib -lmysqlclient udf_exemple.def
mv exemple.dll exemple.udf
```

Note : A cause des limitations de OS/2, les noms des modules UDF ne doivent pas dèpasser 8 caractéres. Les modules sont stockès dans le rèpertoire '`/mysql2/udf`'; le script `safe-mysqld.cmd` placera ce rèpertoire dans la variable d'environnement `BEGINLIBPATH`. Lors de l'utilisation des modules UDF, les extensions spècifiées sont ignorées. Elle est supposée être '`.udf`'. Par exemple, sous Unix, le module partagè peut se nommer '`exemple.so`' et vous chargeriez une de ses fonctions de la façon suivante :

```
mysql> CREATE FUNCTION metaphon RETURNS STRING SONAME "exemple.so";
```

Sous OS/2, me module s'appelera '`exemple.udf`', mais vous n'aurez pas á spècifier son extension :

```
mysql> CREATE FUNCTION metaphon RETURNS STRING SONAME "exemple";
```

2.6.8 Notes relatives á BeOS

Nous sommes vraiment intèressès par le port de MySQL sur BeOS, mais malheureusement, nous n'avons personne qui s'y connaisse en BeOS ou qui ait le temps de s'en occuper.

Nous sommes intèressès par quelqu'un qui serait prêt á faire le port, et nous l'aiderions pour toutes les questions techniques qu'il pourrait se poser durant le processus.

Nous avons dèjá eu des contacts avec des dèveloppeurs BeOS qui ont dit que MySQL ètait portè á 80% sur BeOS, mais nous n'avons plus entendu parler d'eux depuis.

2.6.9 Notes relatives á Novell NetWare

Nous sommes vraiment intèressès par le port de MySQL sur NetWare, mais malheureusement, nous n'avons personne qui s'y connaisse en NetWar ou qui ait le temps de s'en occuper.

Nous sommes intèressès par quelqu'un qui serait prêt á faire le port, et nous l'aiderions pour toutes les questions techniques qu'il pourrait se poser durant le processus.

2.7 Commentaires sur l'installation de Perl

2.7.1 Installer Perl sur Unix

Le support Perl pour MySQL est fournit par les interfaces clientes `DBI/DBD`. Voir Section 8.2 [Perl], page 591. Le code client des modules `DBD/DBI` de Perl, requiére une version de celui-ci supèrieure ou ègale á la version 5.004. L'interface **ne fonctionnera pas** si vous avez une ancienne version de Perl.

Le support Perl de MySQL requiére aussi que vous ayez installè le support de programmation de clients pour MySQL. Si vous avez installè MySQL á partir de fichiers RPM, les programmes clients sont dans le RPM client, mais le support de la programmation de clients est dans le RPM des dèveloppeurs. Assurez-vous d'avoir installè le dernier RPM.

Depuis la version 3.22.8, le support Perl est distribuè sèparèment du reste de la distribution MySQL. Si vous voulez installer le support Perl, les fichiers dont vous aurez besoin peuvent être trouvès sur `http://www.mysql.com/Downloads/Contrib/`.

Les distributions de Perl sont fournies en tant qu'archives comprèssèes `tar` et ont des noms comme '`MODULE-VERSION.tar.gz`', oú `MODULE` est le nom du module et `VERSION` est le numèro de version. Vous devez obtenir les distributions `Data-Dumper`, `DBI`, et `Msql-Mysql-modules` et les installer dans cet ordre. La procèdure d'installation est dècrite ici. L'exemple montrè ici concerne le module `Data-Dumper`, mais la procèdure est la même pour les trois distributions :

1. Dècompressez la distribution dans le dossier courant :

   ```
   shell> gunzip < Data-Dumper-VERSION.tar.gz | tar xvf -
   ```

   Cette commande crèe un dossier appelè '`Data-Dumper-VERSION`'.
2. Mettez vous dans le rèpertoire racine de la distribution dècompressèe :

   ```
   shell> cd Data-Dumper-VERSION
   ```
3. Construisez la distribution et compilez tout :

   ```
   shell> perl Makefile.PL
   shell> make
   shell> make test
   shell> make install
   ```

La commande `make test` est importante car elle vèrifie que le module fonctionne. Notez que quand vous exècutez cette commande durant l'installation de `Msql-Mysql-modules` pour tester le code de l'interface, le serveur MySQL doit être en marche sinon le test èchouera.

Il est bon de reconstruire et rèinstaller la distribution `Msql-Mysql-modules` á chaque fois que vous rèinstallez une nouvelle version de MySQL, particuliérement si vous avez des problémes avec vos scripts `DBI` aprés avoir avoir mis á jour MySQL.

Si vous n'avez pas le droit d'installer des modules Perl dans le dossier systéme ou que vous voulez installer des modules locaux de Perl, la rèfèrence suivante pourra vous aider :

`http://www.iserver.com/support/contrib/perl5/modules.html`

Regardez le paragraphe `Installing New Modules that Require Locally Installed Modules`.

2.7.2 Installer ActiveState Perl sur Windows

Pour installer le module DBD MySQL avec ActiveState Perl sous Windows, vous devez faire ce qui suit :

- Obtenez ActiveState Perl á partir de http://www.activestate.com/Products/ActivePerl/ et installez le.
- Ouvrez un terminal DOS.
- Si requis, dèfinissez la variable HTTP_proxy. Par exemple, vous pouvez faire :

```
set HTTP_proxy=my.proxy.com:3128
```

- Dèmarrez le programme PPM :

```
C:\> c:\perl\bin\ppm.pl
```

- Installez DBI, si ce n'est pas dèjá fait :

```
ppm> install DBI
```

- Si cela fonctionne, exècutez la commande suivante :

```
install \
ftp://ftp.de.uu.net/pub/CPAN/authors/id/JWIED/DBD-mysql-1.2212.x86.ppd
```

Ce qui suit devrait fonctionner avec la version 5.6 d'ActiveState Perl.

Si ce qui prècéde ne veut pas fonctionner, vous devez á la place installer le pilote MyODBC et vous connecter au serveur MySQL via ODBC :

```
use DBI;
$dbh= DBI->connect("DBI:ODBC:$dsn","$utilisateur","$motdepasse") ||
  die "Obtenu l'erreur $DBI::errstr lors de la connexion á $dsn\n";
```

2.7.3 Installer la distribution Perl de MySQL sous Windows

Cette distribution Perl de MySQL contient DBI, DBD:MySQL et DBD:ODBC.

- Obtenez la distribution Perl pour Windows á partir de http://www.mysql.com/downloads/os-win32.
- Dècompressez la distribution dans C: pour obtenir le dossier 'C:\PERL'.
- Ajoutez le rèpertoire 'C:\PERL\BIN' á votre path.
- Ajoutez le rèpertoire 'C:\PERL\BIN\MSWIN32-x86-thread' ou 'C:\PERL\BIN\MSWIN32-x86' á votre path.
- Vèrifiez que perl fonctionne en exècutant perl -v dans un terminal DOS.

2.7.4 Problémes lors de l'utilisation des interfaces Perl DBI et DBD

Si Perl vous informe qu'il ne peut trouver le module '../mysql/mysql.so', il se trouve probablement que Perl n'arrive pas á trouver la librairie partagèe 'libmysqlclient.so'.

Vous pouvez corriger cela en suivant l'une des mèthodes suivantes :

- Compilez la distribution Msql-Mysql-modules avec perl Makefile.PL -static -config au lieu de perl Makefile.PL.
- Copiez 'libmysqlclient.so' dans le dossier oú se situent vos autres librairies partagèes (souvent '/usr/lib' ou '/lib').

- Sous Linux vous pouvez ajouter le chemin vers le dossier dans lequel se trouve '`libmysqlclient.so`' au fichier '`/etc/ld.so.conf`'.
- Ajoutez le chemin complet vers le dossier oú se situe '`libmysqlclient.so`' á la variable d'environnement `LD_RUN_PATH`.

Si vous obtenez l'erreur suivante de `DBD-mysql`, vous utilisez probablement `gcc` (ou un vieux binaire compilè avec `gcc`) :

```
/usr/bin/perl: can't resolve symbol '__moddi3'
/usr/bin/perl: can't resolve symbol '__divdi3'
```

Ajoutez `-L/usr/lib/gcc-lib/... -lgcc` á la commande de liaison lorsque la librairie '`mysql.so`' est construite (vèrifiez l'affichage de `make` concernant '`mysql.so`' quand vous compilez le client Perl). L'option `-L` doit spècifier le chemin vers le dossier oú se situe '`libgcc.a`' sur votre systéme.

Une autre cause du probléme peut être que Perl et MySQL ne sont pas tous deux compilès avec `gcc`. Dans ce cas lá, vous devrez faire en sorte qu'ils le soient.

Si vous obtenez les erreurs suivantes de la part de `Msql-Mysql-modules` quand vous exècutez ces tests :

```
t/00base.............install_driver(mysql) failed:
Can't load '../blib/arch/auto/DBD/mysql/mysql.so' for module DBD::mysql:
../blib/arch/auto/DBD/mysql/mysql.so: undefined symbol:
uncompress at /usr/lib/perl5/5.00503/i586-linux/DynaLoader.pm line 169.
```

cela signifie que vous avez besoin d'inclure la librairie dynamique, -lz, dans la ligne de liaison. Cela peut se faire en changeant ce qui suit dans '`lib/DBD/mysql/Install.pm`' :

```
$sysliblist .= " -lm";

en

$sysliblist .= " -lm -lz";
```

Aprés cela, vous **devez** exècuter 'make realclean' et reprendre l'installation dés le dèbut.

Si vous voulez utiliser le module de Perl sur un systéme qui ne supporte pas les liaisons dynamiques (comme Caldera/SCO) vous pouvez gènèrer une version statique de Perl incluant `DBI` et `DBD-mysql`. L'approche est de gènèrer une version de Perl avec le code de `DBI` liè et de l'installer au dessus de votre Perl courant. Puis vous utilisez cette version pour en crèer á nouveau une autre qui comporte le code de `DBD` liè et d'installer cette version ci.

Sur Caldera (SCO), vous devez dèfinir les variables d'environnement suivantes :

```
shell> LD_LIBRARY_PATH=/lib:/usr/lib:/usr/local/lib:/usr/progressive/lib
ou
shell> LD_LIBRARY_PATH=/usr/lib:/lib:/usr/local/lib:/usr/ccs/lib:\
/usr/progressive/lib:/usr/skunk/lib
shell> LIBPATH=/usr/lib:/lib:/usr/local/lib:/usr/ccs/lib:\
/usr/progressive/lib:/usr/skunk/lib
shell> MANPATH=scohelp:/usr/man:/usr/local1/man:/usr/local/man:\
/usr/skunk/man:
```

D'abord, crèez un Perl incluant un `DBI` liè statiquement en exècutant des commandes dans le dossier oú se situe votre distribution `DBI` :

```
shell> perl Makefile.PL -static -config
shell> make
shell> make install
shell> make perl
```

Ensuite, vous devez installer le nouveau Perl. Les affichages de `make perl` vous indiqueront les commandes `make` exactes que vous aurez besoin d'exècuter pour faire l'installation. Sur Caldera (SCO), il s'agit de `make -f Makefile.aperl inst_perl MAP_TARGET=perl`.

Puis, utilisè le Perl qui vient d'être crèè pour en crèer un nouveau qui inclut un `DBD::mysql` liè statiquement en exècutant ces commandes dans le dossier oú votre distribution de `Msql-Mysql-modules` se situe :

```
shell> perl Makefile.PL -static -config
shell> make
shell> make install
shell> make perl
```

Finalement, vous devez installer ce nouveau Perl. Une fois de plus, l'affichage de `make perl` vous indiquera la commande á utiliser.

3 Tutoriels d'introduction

Ce chapitre fournit un tutoriel d'introduction á MySQL en montrant comment utiliser le client `mysql` pour crèer et utiliser une simple base de donnèes. `mysql` (quelques fois nommè "moniteur terminal" ou juste "moniteur") est un programme interactif qui vous permet de vous connecter á un serveur MySQL, exècuter des requêtes et voir les rèsultats. `mysql` peut aussi être utilisè en mode batch : vous placez vos requêtes dans un fichier, puis vous faites exècuter á `mysql` le contenu de ce fichier. Les deux maniéres d'utiliser `mysql` sont expliquèes ici.

Pour voir une liste d'options fournies par `mysql`, invoquez-le avec l'option `--help` :

```
shell> mysql --help
```

Ce chapitre assume que `mysql` est installè sur votre machine et qu'un serveur MySQL est disponible pour que vous vous y connectiez. Si ce n'est pas le cas, contactez votre administrateur MySQL. (Si **vous** êtes l'administrateur, vous aurez besoin de consulter d'autres sections de ce manuel.)

Ce chapitre dècrit le processus d'installation et d'utilisation d'une base de donnèes en entier. Si vous n'êtes intèressès que par l'accés á une base de donnèes existante, vous pouvez sauter les sections dècrivant la crèation de la base et des tables.

Ce chapitre n'est qu'un tutoriel, beaucoup de dètails ne sont pas approfondis. Consultez les sections appropriées du manuel pour plus d'informations sur les sujets abordès.

3.1 Connexion et dèconnexion au serveur

Pour vous connecter au serveur, vous aurez dans la plupart des cas á fournir un nom d'utilisateur á MySQL, et, sûrement, un mot de passe. Si le serveur fonctionne sur une autre machine que la vôtre, vous devrez spècifier son adresse. Contactez votre administrateur pour connaître les paramétres á utiliser lors de la connexion (hôte, nom d'utilisateur, mot de passe á utiliser...). Une fois que vous aurez les bons paramétres, vous pourrez vous connecter de la façon suivante :

```
shell> mysql -h hote -u utilisateur -p
Enter password: ********
```

`********` reprèsente votre mot de passe, entrez-le lorsque `mysql` affiche `Enter password:`.

Si tout fonctionne, vous devrez voir quelques informations d'introduction suivies d'une invite de commande `mysql>` :

```
shell> mysql -h host -u user -p
Enter password: ********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 459 to server version: 3.22.20a-log

Type 'help' for help.

mysql>
```

L'invite vous dit que `mysql` attend que vous entriez des commandes.

Quelques installations de MySQL autorisent les connexions anonymes au serveur tournant sur l'hôte local. Si c'est la cas sur votre machine, vous devriez arriver á vous connecter á ce serveur en invoquant la commande `mysql` sans aucune option :

```
shell> mysql
```

Aprés vous être connectè avec succés, vous pouvez vous dèconnecter á tout moment en entrant `QUIT` dans l'invite `mysql>` :

```
mysql> QUIT
Bye
```

Vous pouvez aussi le faire en appuyant sur Ctrl-D.

La plupart des exemples dans les sections suivantes supposent que vous êtes connectè au serveur. Cela se voit á l'invite `mysql>`.

3.2 Entrer des requêtes

Assurez-vous d'être connectè au serveur, comme expliquè prècèdemment dans cette section. Faire ceci ne sèlectionnera pas une base par lui même, mais c'est normal. A ce stade, il est important de dècouvrir la façon dont sont publiées les requêtes, pour ensuite pouvoir crèer des tables, y insèrer et rechercher des donnèes. Cette section dècrit les principes de base pour entrer une commande, en utilisant plusieurs requêtes que vous pouvez essayer pour vous familiariser avec la façon dont `mysql` fonctionne.

Voilá une commande simple qui demande au serveur de vous donner son numèro de version et la date courante. Entrez-la comme suit, juste aprés l'invite `mysql>` puis pressez Enter :

```
mysql> SELECT VERSION(), CURRENT_DATE;
+--------------+--------------+
| VERSION()    | CURRENT_DATE |
+--------------+--------------+
| 3.22.20a-log | 1999-03-19   |
+--------------+--------------+
1 row in set (0.01 sec)
mysql>
```

La requête rèvéle plusieurs choses á propos de `mysql` :

- Une commande consiste normalement en une commande SQL suivie d'un point-virgule. (Il y'a quelques cas ou le point-virgule n'est pas requis. `QUIT`, mentionnèe plus tôt, en fait partie. Nous verrons les autres plus tard.)
- Lorsque vous entrez une commande, `mysql` l'envoi au serveur pour l'exècution et affiche le rèsultat, puis affiche un autre `mysql>` pour indiquer qu'il attend une autre commande.
- `mysql` affiche le rèsultat des requêtes dans une table (lignes et colonnes). La premiére ligne contient le nom des colonnes. Les lignes suivantes constituent le rèsultat de la requête. Normalement, les titres des colonnes sont les noms des champs des tables de la base de donnèes que vous avez rècupèrès. Si vous rècupèrez la valeur d'une expression au lieu d'une colonne (comme dans l'exemple prècèdent), `mysql` nomme la colonne en utilisant l'expression elle-même.
- `mysql` vous indique combien de lignes ont ètè retournèes et combien de temps d'exècution la requête a pris, ce qui vous donnera une approximation des performances

du serveur. Ces valeurs sont imprècises car elles reprèsentent le temps soft (et non le temps CPU ou hard), et qu'elles sont affectèes par des facteurs tels que la charge du serveur ou l'accessibilitè du rèseau. (Dans un soucis de briévetè, la ligne contenant "rows in set" n'est plus montrèe dans les exemples suivants de ce chapitre.)

Les mots-clefs peuvent être entrès sous n'importe quelle forme de casse. Les requêtes suivantes sont èquivalentes :

```
mysql> SELECT VERSION(), CURRENT_DATE;
mysql> select version(), current_date;
mysql> SeLeCt vErSiOn(), current_DATE;
```

Voilá une autre requête. Elle montre que vous pouvez utiliser `mysql` en tant que simple calculatrice :

```
mysql> SELECT SIN(PI()/4), (4+1)*5;
+------------+---------+
| SIN(PI()/4) | (4+1)*5 |
+-------------+---------+
|    0.707107 |      25 |
+-------------+---------+
```

Les commandes vues jusqu'á prèsent ont ètè relativement courtes, et tenaient sur une seule ligne. Vous pouvez même entrer plusieurs requêtes sur une seule ligne. Il suffit de terminer chacune d'elle par un point-virgule :

```
mysql> SELECT VERSION(); SELECT NOW();
+--------------+
| VERSION()    |
+--------------+
| 3.22.20a-log |
+--------------+

+---------------------+
| NOW()               |
+---------------------+
| 1999-03-19 00:15:33 |
+---------------------+
```

Une commande ne doit pas être obligatoirement sur une seule ligne ; les commandes qui exigent plusieurs lignes ne sont pas un probléme. `mysql` dètermine oú se situe la fin de votre commande en recherchant le point-virgule de terminaison, et pas l'extrèmitè de la commande entrèe. (Dans d'autres termes, `mysql` accepte des formats libres d'entrèe : il collecte les lignes entrèes mais ne les exècute qu'une fois le point-virgule trouvè.)

Voilá une seule requête sur plusieurs lignes :

```
mysql> SELECT
    -> USER()
    -> ,
    -> CURRENT_DATE;
+--------------------+--------------+
| USER()             | CURRENT_DATE |
+--------------------+--------------+
```

```
| joesmith@localhost | 1999-03-18   |
+--------------------+--------------+
```

Dans cet exemple, notez comment l'invite change de `mysql>` á `->` aprés avoir entrè la premiére ligne d'une requête multi-lignes. C'est la façon dont `mysql` indique qu'il n'a pas vu de requête compléte et qu'il attend la fin de celle-ci. L'invite est votre ami en vous fournissant la rètroactivitè. Si vous utilisez cette rètroactivitè, vous vous rendrez toujours compte de ce que `mysql` attend.

Si vous dècidez d'annuler une commande que vous êtes en train de taper, faites-le en entrant `\c` :

```
mysql> SELECT
    -> USER()
    -> \c
mysql>
```

Ici aussi, portez votre attention sur l'invite. Elle se transforme á nouveau en `mysql>` aprés que vous ayez entrè `\c`, vous informant que `mysql` est prêt pour une nouvelle requête.

Le tableau suivant montre les diffèrentes invites que vous pourrez voir et rèsume leur signification quand á l'ètat dans lequel se trouve `mysql` :

Invite	**Signification**
`mysql>`	Prêt pour une nouvelle commande.
`->`	En attente de la ou des lignes terminant la commande.
`'>`	En attente de la prochaine ligne collectant une chaîne commencèe par un guillemet simple (‘'’).
`">`	En attente de la prochaine ligne collectant une chaîne commencèe par un guillemet double (‘"’).

Les commandes sur plusieurs lignes sont la plupart du temps des accidents, lorsque vous voulez faire une commande sur une seule ligne et que vous oubliez le point-virgule de fin. Dans ce cas, `mysql` attend la suite de votre saisie :

```
mysql> SELECT USER()
    ->
```

Si cela vous arrive (vous pensez que votre requête est compléte mais la seule rèponse est l'invite `->`), il est fort probable que `mysql` attende le point-virgule. Si vous ne notez pas ce que l'invite vous indique, vous pourriez patienter pendant longtemps avant de rèaliser ce que vous devez faire. Entrez un point-virgule pour complèter la requête, et `mysql` devrait l'exècuter :

```
mysql> SELECT USER()
    -> ;
+--------------------+
| USER()             |
+--------------------+
| joesmith@localhost |
+--------------------+
```

L'invite `'>` ainsi que `">` apparaissent durant l'entrèe de chaîne. Dans MySQL, vous pouvez ècrire une chaîne entourèe du caractére ‘'’ ou bien ‘"’ (par exemple, `'Bonjour'` or `"Au Revoir"`), et `mysql` vous laisse entrer une chaîne qui peut être sur plusieurs lignes. Lorsque

vous voyez une invite comme `'>` ou `">`, cela signifie que vous avez entrè une ligne contenant le caractére `'` ou `"`, mais vous n'avez pas encore entrè le caractére correspondant qui termine votre chaîne. C'est pratique si vous entrez rèellement une chaîne á lignes multiples, mais est-ce probable ? Pas vraiment. Plus souvent, les invites `'>` et `">` indiquent que vous avez, par inadvertance, oubliè un caractére de fermeture. Par exemple :

```
mysql> SELECT * FROM ma_table WHERE nom = "Smith AND age < 30;
    ">
```

Si vous entrez cette requête `SELECT`, puis appuyez sur Enter et attendez le rèsultat, rien ne se passera. Au lieu de vous demander pourquoi la requête met si longtemps á s'exècuter, remarquez que l'invite de commande s'est transformèe en `">`. Cela indique que `mysql` attend de voir la fin d'une chaîne de caractéres non-terminèe. (Voyez-vous l'erreur dans cette requête ? Il manque le second guillemet á la suite de `"Smith`.)

Que faire ? Le plus simple est d'annuler la commande. Toutefois, vous ne pouvez vous contenter de taper `\c` dans ce cas-lá, car `mysql` l'interpréte comme une partie de la chaîne qu'il est en train de collecter ! A la place, entrez le second guillemet (pour que `mysql` sache que vous avez fini la chaîne), puis entrez `\c` :

```
mysql> SELECT * FROM my_table WHERE name = "Smith AND age < 30;
    "> "\c
mysql>
```

L'invite se change á nouveau en `mysql>`, indiquant que `mysql` est prêt pour une nouvelle requête.

Il est important de savoir ce que les invites `'>` et `">` signifient, car si vous avez entrè par erreur une chaîne non terminèe, toutes les lignes suivantes que vous entrerez seront ignorèes par `mysql`, même une ligne contenant `QUIT` ! Cela peut prêter á confusion, spècialement si vous ne savez pas que vous devez fournir le guillemet fermant avant de pouvoir annuler la commande courante.

3.3 Crèation et utilisation d'une base de donnèes

Maintenant que vous savez entrer des commandes, il est temps d'accèder á une base.

Supposons que vous avec plusieurs animaux chez vous (dans votre mènagerie) et que vous voulez garder diverses informations les concernant. Vous pouvez le faire en crèant des tables pour stocker vos donnèes et y charger vos informations. Vous pourrez alors rèpondre á diffèrentes sortes de questions á propos de vos animaux en rècupèrant les donnèes á partir des tables. Cette section vous montre comment :

- Crèer une base de donnèes
- Crèer une table
- Charger des donnèes dans vos tables
- Rècupèrer des donneèes á partir des tables de diffèrentes façons
- Utiliser plusieurs tables

La base de donnèes de la mènagerie va être simple (dèlibèrèment), mais il n'est pas difficile de penser á des situations courantes de la vie oú vous aurez á utiliser un tel type de base de donnèes. Par exemple, une telle base pourrait être utilisèe

par un èleveur pour gèrer sa boutique, ou par un vètèrinaire pour garder des traces de ses patients. Une distribution de la mènagerie contenant quelques requêtes et des exemples de données utilisèes dans la section suivante peuvent être trouvès sur le site web de MySQL. Ils sont disponibles au format compressè `tar` (`http://www.mysql.com/Downloads/Contrib/Examples/menagerie.tar.gz`) ou au format Zip (`http://www.mysql.com/Downloads/Contrib/Examples/menagerie.zip`).

Utilisez la commande `SHOW` pour trouver quelles bases existent dèjá sur le serveur :

```
mysql> SHOW DATABASES;
+----------+
| Database |
+----------+
| mysql    |
| test     |
| tmp      |
+----------+
```

La liste des bases de donnèes est probablement diffèrente sur votre machine, mais les bases `mysql` et `test` y figurent sûrement. La base `mysql` est requise car elle gére les accés et les priviléges. La base `test` est souvent fournie pour que les utilisateurs y effectuent leurs tests.

Notez que vous ne pourrez voir toutes les bases de donnèes si vous n'avez pas le privilége `SHOW DATABASES`. Voir Section 4.3.1 [GRANT], page 226.

Si la base de donnèes `test` existe, essayez d'y accèder :

```
mysql> USE test
Database changed
```

Notez que `USE`, comme `QUIT`, ne requiert pas de point-virgule. (Vous pouvez terminer ces commandes avec un point-virgule ; cela ne posera pas de problémes.) La commande `USE` est spèciale d'un autre point de vue : elle doit être donnèe sur une seule ligne.

Vous pouvez utiliser la base de donnèes `test` (si vous y avez accés) pour les exemples qui suivent, mais tout ce que vous crèerez dans cette base pourra être èffacè par quiconque y a accés. Pour cette raison, vous feriez mieux de demander á votre administrateur MySQL la permission d'utiliser une base de donnèes rien que pour vous. Supposez que vous voulez nommer la votre `menagerie`. L'administrateur a besoin d'exècuter une commande telle que :

```
mysql> GRANT ALL ON menagerie.* TO votre_nom_mysql;
```

oú `votre_nom_mysql` est le nom d'utilisateur MySQL qui vous est assignè.

3.3.1 Crèer et sèlectionner une base de donnèes

Si l'administrateur vous a crèè une base de donnèes lors du paramètrage de vos droits, vous pouvez commencer á l'utiliser. Sinon, vous aurez besoin de la crèer par vous-même :

```
mysql> CREATE DATABASE menagerie;
```

Sous Unix, les noms des bases de donnèes sont sensibles á la casse (ce qui différe des mots rèservès de SQL), ce qui fait que vous devez tuojours vous rèfèrer á votre base de donnèes avec `menagerie`, non avec `Menagerie`, `MENAGERIE`, ou d'autres variantes. Cela est aussi valable pour les noms de tables. (Sous Windows, cette restriction n'est pas appliquèe,

même si vous devez vous rèfèrer á une table ou une base de la même façon dans une même requête).

La crèation d'une base de donnèes ne la sèlectionne pas pour l'utilisation ; vous devez le faire explicitement. Pour rendre `menagerie` la base courante, utilisez cette commande :

```
mysql> USE menagerie
Database changed
```

Votre base a besoin d'être crèèe juste une fois, mais vous devez la sèlectionner pour l'utiliser, chaque fois que vous dèbutez une session `mysql`. Vous pouvez le faire en publiant une requête `USE` comme ci-dessus. Sinon, vous pouvez sèlectionner la base directement dans la ligne de commande lorsque vous invoquez `mysql`. Vous devez juste spècifier son nom aprés les paramétres de connexion dont vous avez besoin. Par exemple :

```
shell> mysql -h hote -u utilisateur -p menagerie
Enter password: ********
```

Notez que `menagerie` n'est pas votre mot de passe dans la commande que nous venons de montrer. Si vous voulez le fournir dans la ligne de commande aprés l'option `-p`, vous devez le faire sans espace entre les deux (par exemple, tapez `-pmonmotdepasse`, et non `-p monmotdepasse`). Toutefois, mettre le mot de passe en ligne de commande n'est pas recommandè, car le faire permettrait á d'autres utilisateurs connectès sur votre machine de l'obtenir.

3.3.2 Crèation d'une table

Crèer la base de donnèes est la partie facile, mais jusque-lá elle est vide, comme vous le montre `SHOW TABLES` :

```
mysql> SHOW TABLES;
Empty set (0.00 sec)
```

La partie la plus difficile est le choix de la structure de la base de donnèes : de quelles tables aurez vous besoin et quelles colonnes devront figurer dans chacune d'elles.

Vous voudrez une table qui contient un enregistrement pour chaque animal. On peut l'appeler la table `animal`, et elle devra contenir, au minimum, le nom de chaque animal. Puisque le nom tout seul n'est pas intèressant, la table devra contenir d'autres informations. Par exemple, si plus d'une personne de votre famille posséde un animal, vous voudrez lister le nom du maître de chaque animal. Vous voudrez peut-être aussi enregistrer une description basique comme l'espéce ou le sexe.

Et pour l'âge ? C'est intèressant, mais n'est pas bon pour un stockage en base de donnèes. L'âge change chaque jour, vous devrez donc mettre á jour vos enregistrements assez souvent. Il est prèfèrable de stocker une valeur fixe, comme la date de naissance. Dans ce cas-lá, á chaque fois que vous aurez besoin de l'âge, vous pourrez l'obtenir en faisant la diffèrence entre la date courante et la date enregistrèe. MySQL fournit des fonctions de calcul sur les dates, cela ne sera donc pas difficile. Enregistrer la date de naissance, au lieu de l'âge a d'autres avantages :

- Vous pouvez utiliser la base de donnèes pour des tâches, comme la gènèration d'un rappel pour les prochains anniversaires d'animaux. (Si vous trouvez que ce type de requêtes est quelque peu idiot, notez que c'est la même question que vous vous poseriez

dans le contexte d'une base de donnèes d'affaires pour identifier les clients á qui vous aurez besoin d'envoyer un message de voeux, pour cette touche informatiquement assistèe d'humanisme.)

- Vous pouvez calculer l'âge á partir d'autres dates que la date du jour. Par exemple, si vous stockez la date de la mort dans la base de donnèes, vous pourrez facilement calculer l'âge qu'avait un animal á sa mort.

Vous trouverez probablement d'autres informations qui pourront être utiles dans la table `animal`, mais celles identifiès jusqu'á maintenant sont largement suffisantes pour l'instant : nom, maitre, espece, sexe, naissance, et mort.

Utilisez une requête `CREATE TABLE` pour spècifier la structure de votre table :

```
mysql> CREATE TABLE animal (nom VARCHAR(20), maitre VARCHAR(20),
    -> espece VARCHAR(20), sexe CHAR(1), naissance DATE, mort DATE);
```

`VARCHAR` est un bon choix pour les colonnes `nom`, `maitre`, et `espece` car leurs valeurs varient en longueur. La longueur de ces colonnes ne doit pas nècessairement être la même, et n'a pas besoin d'être forcement `20`. Vous pouvez choisir une taille entre `1` et `255`, celle qui vous semblera la plus raisonnable. (Si vous faites un mauvais choix et que vous vous apercevez plus tard que vous avez besoin d'un champ plus long, MySQL fournit la commande `ALTER TABLE`.)

Le sexe des animaux peut être reprèsentè de plusieurs façons, par exemple, `"m"` et `"f"`, ou bien `"male"` and `"femelle"`. Il est plus simple d'utiliser les caractéres simples `"m"` et `"f"`.

L'utilisation du type de donnèes `DATE` pour les colonnes `naissance` et `mort` est un choix plutôt judicieux.

Maintenant que vous avez crèè une table, `SHOW TABLES` devrait produire de l'affichage :

```
mysql> SHOW TABLES;
+----------------------+
| Tables in menagerie |
+----------------------+
| animal               |
+----------------------+
```

Pour vèrifier que la table a ètè crèèe de la façon que vous vouliez , utilisez la commande `DESCRIBE` :

```
mysql> DESCRIBE animal;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| nom       | varchar(20) | YES  |     | NULL    |       |
| maitre    | varchar(20) | YES  |     | NULL    |       |
| espece    | varchar(20) | YES  |     | NULL    |       |
| sexe      | char(1)     | YES  |     | NULL    |       |
| naissance | date        | YES  |     | NULL    |       |
| mort      | date        | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
```

Vous pouvez utiliser `DESCRIBE` quand vous voulez, par exemple, si vous avez oubliè les noms des colonnes dans votre table ou leurs types.

3.3.3 Charger des donnèes dans une table

Aprés la crèation de votre table, vous aurez besoin de la remplir. Les commandes `LOAD DATA` et `INSERT` sont utiles pour cela.

Supposons que les enregistrements de vos animaux peuvent être dècrits comme suit. (Observez que MySQL attend les dates au format `YYYY-MM-DD`; cela peut diffèrer de ce á quoi vous êtes habituè.)

nom	maitre	especes	sexe	naissance	mort
Fluffy	Harold	chat	f	1993-02-04	
Claws	Gwen	chat	m	1994-03-17	
Buffy	Harold	chien	f	1989-05-13	
Fang	Benny	chien	m	1990-08-27	
Bowser	Diane	chien	m	1998-08-31	1995-07-29
Chirpy	Gwen	oiseau	f	1998-09-11	
Whistler	Gwen	oiseau		1997-12-09	
Slim	Benny	serpent	m	1996-04-29	

Puisque vous commencez avec une table vide, il est facile de la remplir en crèant un fichier texte contenant une ligne pour chaque animal que vous avez, puis charger son contenu á l'aide d'une seule commande.

Vous pouvez crèer un fichier '`pet.txt`' contenant un enregistrement par ligne, avec les valeurs sèparès par des tabulations, et ordonnèes comme les champs l'ètaient dans la requête `CREATE TABLE`. Pour les donnèes manquantes (comme un sexe inconnu ou la date de mort d'un animal toujours en vie), vous pouvez utiliser les valeurs `NULL`. Pour les reprèsenter dans votre fichier texte, utilisez `\N`. Par exemple, l'enregistrement de Whistler l'oiseau ressemblera á ça (l'espace entre les valeurs est une tabulation) :

```
nom       maitre  especes  sexe  naissance  mort
Whistler  Gwen    bird     \N    1997-12-   \N
                                 09
```

Pour charger le fichier '`pet.txt`' dans la table `pet`, utilisez cette commande :

```
mysql> LOAD DATA LOCAL INFILE "pet.txt" INTO TABLE pet;
```

Vous pouvez spècifier la valeur du sèparateur de colonnes et le marqueur de fin de lignes explicitement dans la commande `LOAD DATA` si vous le voulez, mais les valeurs par dèfaut sont la tabulation et le retour á la ligne. Ceux-lá sont suffisants pour que la commande lise le fichier '`pet.txt`' correctement.

Lorsque vous voulez ajouter des enregistrements un par un, la commande `INSERT` est utile. Dans sa forme la plus simple, oú vous spècifiez une valeur pour chaque colonne, dans l'ordre oú les colonnes sont listèes dans la requête `CREATE TABLE`. Supposons que Diane achéte un nouvel hamster nommè Puffball. Vous pourriez ajouter ce nouvel enregistrement en utilisant un `INSERT` de la façon suivante :

```
mysql> INSERT INTO pet
    -> VALUES ('Puffball','Diane','hamster','f','1999-03-30',NULL);
```

Notez que les chaînes de caractéres et les valeurs de dates sont spècifièes en tant que chaînes protègèes par des guillemets. De plus, avec `INSERT` vous pouvez insèrer la valeur `NULL` directement pour reprèsenter une valeur manquante. Vous n'utilisez pas `\N` comme vous le faites avec `LOAD DATA`.

A partir de cet exemple, vous devriez être capable de voir qu'il y'a beaucoup plus de commandes á taper lorsque vous utilisez la commande `INSERT` au lieu de `LOAD DATA`.

3.3.4 Rècupèrer des informations á partir d'une table

La commande `SELECT` est utilisèe pour rècupèrer des informations á partir d'une table. La forme usuelle est :

```
SELECT quoi_selectionner
FROM quel_table
WHERE conditions_a_satisfaire
```

`quoi_selectionner` indique ce que vous voulez voir. Cela peut être une liste de colonnes, ou `*` pour indiquer "toutes les colonnes". `quel_table` indique la table á partir de laquelle rècupèrer les donnèes. La clause `WHERE` est optionnelle. Si elle est prèsente, `conditions_a_satisfaire` spècifie les conditions que les lignes doivent satisfaire pour être sèlèctionnèes.

3.3.4.1 Sèlectionner toutes les donnèes

La plus simple forme de `SELECT` rècupére toutes les donnèes d'une table :

```
mysql> SELECT * FROM animal;
+----------+--------+---------+------+------------+------------+
| nom      | maitre | espece  | sexe | naissance  | mort       |
+----------+--------+---------+------+------------+------------+
| Fluffy   | Harold | chat    | f    | 1993-02-04 | NULL       |
| Claws    | Gwen   | chat    | m    | 1994-03-17 | NULL       |
| Buffy    | Harold | chien   | f    | 1989-05-13 | NULL       |
| Fang     | Benny  | chien   | m    | 1990-08-27 | NULL       |
| Bowser   | Diane  | chien   | m    | 1998-08-31 | 1995-07-29 |
| Chirpy   | Gwen   | oiseau  | f    | 1998-09-11 | NULL       |
| Whistler | Gwen   | oiseau  | NULL | 1997-12-09 | NULL       |
| Slim     | Benny  | serpent | m    | 1996-04-29 | NULL       |
| Puffball | Diane  | hamster | f    | 1999-03-30 | NULL       |
+----------+--------+---------+------+------------+------------+
```

Cette forme de `SELECT` est utile si vous voulez rècupèrer la table entiére. par exemple, aprés l'avoir juste remplie avec vos donnèes d'origine. Il apparaît alors qu'une erreur s'ètait glissèe dans votre fichier de donnèes : Bowser a l'air d'être nè aprés sa mort ! En consultant le papier original de son pedigree, vous trouvez que la date correcte est 1989 et non pas 1998.

Il y'a au moins deux façons de corriger cela :

- Corriger le fichier '`animal.txt`' pour corriger l'erreur, puis vider et recharger á nouveau la table en utilisant `DELETE` et `LOAD DATA` :

  ```
  mysql> SET AUTOCOMMIT=1;  # Utilisè pour une recrèation rapide de la table
  mysql> DELETE FROM animal;
  mysql> LOAD DATA LOCAL INFILE "animal.txt" INTO TABLE animal;
  ```

  Toutefois, si vous choisissez cette mèthode, vous devrez aussi rentrer á nouveau l'enregistrement de Puffball.

- Corriger uniquement l'enregistrement erronè avec une requête `UPDATE` :

```
mysql> UPDATE animal SET naissance = "1989-08-31" WHERE nom = "Bowser";
```

Comme nous l'avons montrè, il est facile de rècupèrer toutes les données d'une table. Toutefois, vous ne voudrez sûrement pas le faire, surtout si la table devient imposante. A la place, vous serez plus intèressè par rèpondre á une question particuliére, dans ce cas-lá, vous spècifiez quelques contraintes pour les informations que vous voulez. Regardons quelques requêtes de sèlection qui rèpondent á des questions á propos de vos animaux.

3.3.4.2 Sèlectionner des lignes particulières

Vous pouvez sèlectionner des lignes particuliéres de votre table. Par exemple, si vous voulez vèrifier la modification que vous avez effectuèe sur la date de naissance de Bowser, sèlectionnez son enregistrement comme suit :

```
mysql> SELECT * FROM animal WHERE nom = "Bowser";
+--------+--------+--------+------+------------+------------+
| nom    | maitre | espece | sexe | naissance  | mort       |
+--------+--------+--------+------+------------+------------+
| Bowser | Diane  | chien  | m    | 1989-08-31 | 1995-07-29 |
+--------+--------+--------+------+------------+------------+
```

L'affichage confirme que la date est correcte maintenant : 1989, et non 1998.

La comparaison des chaînes de caractéres se fait normalement avec sensibilitè á la casse, vous pouvez donc spècifier le nom `"bowser"`, `"BOWSER"`, etc. Le rèsultat de la requête sera le même.

Vous pouvez spècifier des conditions sur toutes les colonnes, pas seulement `nom`. Par exemple, si vous voulez savoir quels animaux sont nès aprés 1998, testez la colonne naissance :

```
mysql> SELECT * FROM animal WHERE naissance >= "1998-1-1";
+----------+--------+---------+------+------------+-------+
| nom      | maitre | espece  | sexe | naissance  | mort  |
+----------+--------+---------+------+------------+-------+
| Chirpy   | Gwen   | oiseau  | f    | 1998-09-11 | NULL  |
| Puffball | Diane  | hamster | f    | 1999-03-30 | NULL  |
+----------+--------+---------+------+------------+-------+
```

Vous pouvez combiner plusieurs conditions, par exemple, pour trouver les chiennes :

```
mysql> SELECT * FROM animal WHERE espece = "chien" AND sexe = "f";
+-------+--------+--------+------+------------+-------+
| nom   | maitre | espece | sexe | naissance  | mort  |
+-------+--------+--------+------+------------+-------+
| Buffy | Harold | chien  | f    | 1989-05-13 | NULL  |
+-------+--------+--------+------+------------+-------+
```

La requête prècèdente utilise l'opèrateur logique `AND`. L'opèrateur `OR` existe aussi :

```
mysql> SELECT * FROM animal WHERE espece = "serpent" OR espece = "oiseau";
+----------+--------+---------+------+------------+-------+
| nom      | maitre | espece  | sexe | naissance  | mort  |
```

```
+----------+--------+---------+------+------------+-------+
| Chirpy   | Gwen   | oiseau  | f    | 1998-09-11 | NULL  |
| Whistler | Gwen   | oiseau  | NULL | 1997-12-09 | NULL  |
| Slim     | Benny  | serpent | m    | 1996-04-29 | NULL  |
+----------+--------+---------+------+------------+-------+
```

`AND` et `OR` peuvent être utilisès ensemble. Si vous le faites, une bonne idèe est d'utiliser les parenthéses pour indiquer comment les conditions doivent être regroupèes :

```
mysql> SELECT * FROM animal WHERE (espece = "chat" AND sexe = "m")
    -> OR (espece = "chien" AND sexe = "f");
+-------+--------+--------+------+------------+-------+
| nom   | maitre | espece | sexe | naissance  | mort  |
+-------+--------+--------+------+------------+-------+
| Claws | Gwen   | chat   | m    | 1994-03-17 | NULL  |
| Buffy | Harold | chien  | f    | 1989-05-13 | NULL  |
+-------+--------+--------+------+------------+-------+
```

3.3.4.3 Sèlectionner des colonnes particuliéres

Si vous ne voulez pas voir les lignes entiéres de votre table, nommez les colonnes qui vous intèressent, en les sèparant par des virgules. Par exemple, si vous voulez savoir quand vos animaux sont nès, sèlectionnez les colonnes `nom` et `naissance` :

```
mysql> SELECT nom, naissance FROM animal;
+----------+------------+
| nom      | naissance  |
+----------+------------+
| Fluffy   | 1993-02-04 |
| Claws    | 1994-03-17 |
| Buffy    | 1989-05-13 |
| Fang     | 1990-08-27 |
| Bowser   | 1989-08-31 |
| Chirpy   | 1998-09-11 |
| Whistler | 1997-12-09 |
| Slim     | 1996-04-29 |
| Puffball | 1999-03-30 |
+----------+------------+
```

Pour trouver qui posséde les animaux, utilisez cette requête :

```
mysql> SELECT maitre FROM animal;
+--------+
| maitre |
+--------+
| Harold |
| Gwen   |
| Harold |
| Benny  |
| Diane  |
```

```
| Gwen   |
| Gwen   |
| Benny  |
| Diane  |
+--------+
```

Toutefois, remarquez que la requête rècupére le champ `maitre` de chaque enregistrement, et certains apparaissent plus d'une fois. Pour minimiser l'affichage, rècupèrez chaque rèsultat unique une seule fois en ajoutant le mot-clef `DISTINCT` :

```
mysql> SELECT DISTINCT maitre FROM animal;
+--------+
| maitre |
+--------+
| Benny  |
| Diane  |
| Gwen   |
| Harold |
+--------+
```

Vous pouvez utiliser une clause `WHERE` pour combiner la sèlection des lignes avec celle des colonnes. Par exemple, pour obtenir les dates de naissance des chiens et chats uniquement, utilisez cette requête :

```
mysql> SELECT nom, espece, naissance FROM animal
    -> WHERE espece = "chien" OR espece = "chat";
+--------+--------+------------+
| nom    | espece | naissance  |
+--------+--------+------------+
| Fluffy | chat   | 1993-02-04 |
| Claws  | chat   | 1994-03-17 |
| Buffy  | chien  | 1989-05-13 |
| Fang   | chien  | 1990-08-27 |
| Bowser | chien  | 1989-08-31 |
+--------+--------+------------+
```

3.3.4.4 Trier les enregistrements

Vous avez sûrement notè dans les exemples prècèdents que les lignes de rèsultat sont affichèes sans ordre particulier. Cependant, il est souvent plus facile d'examiner les rèsultats lorsqu'ils sont triès d'une maniére significative. Pour trier un rèsultat, vous devez utiliser une clause `ORDER BY`.

L'exemple suivant prèsente les dates d'anniversaire des animaux, triées par date :

```
mysql> SELECT nom, naissance FROM animaux ORDER BY naissance;
+----------+------------+
| nom      | naissance  |
+----------+------------+
| Buffy    | 1989-05-13 |
| Bowser   | 1989-08-31 |
```

```
| Fang     | 1990-08-27 |
| Fluffy   | 1993-02-04 |
| Claws    | 1994-03-17 |
| Slim     | 1996-04-29 |
| Whistler | 1997-12-09 |
| Chirpy   | 1998-09-11 |
| Puffball | 1999-03-30 |
+----------+------------+
```

Sur les noms de colonnes, le tri — comme toutes les opèrations de comparaison — est normalement exècutè sans tenir compte de la casse. Cela signifie que l'ordre sera indèfini pour les colonnes qui sont identiques, exceptè leur casse. Vous pouvez forcer le tri sensible á la casse en utilisant la clause BINARY : `ORDER BY BINARY(champ)`.

Pour trier dans l'ordre inverse, ajoutez le mot-clè `DESC` (dècroissant) au nom de la colonne á trier :

```
mysql> SELECT nom, naissance FROM animaux ORDER BY naissance DESC;
+----------+------------+
| nom      | naissance  |
+----------+------------+
| Puffball | 1999-03-30 |
| Chirpy   | 1998-09-11 |
| Whistler | 1997-12-09 |
| Slim     | 1996-04-29 |
| Claws    | 1994-03-17 |
| Fluffy   | 1993-02-04 |
| Fang     | 1990-08-27 |
| Bowser   | 1989-08-31 |
| Buffy    | 1989-05-13 |
+----------+------------+
```

Vous pouvez effectuer un tri sur plusieurs colonnes. Par exemple, pour trier par types d'animaux, puis par la date d'anniversaire des animaux, en plaçant les plus jeunes en premier, utilisez la requête suivante :

```
mysql> SELECT nom, espece, naissance FROM animaux ORDER BY type, naissance DESC;
+----------+---------+------------+
| nom      | espece  | naissance  |
+----------+---------+------------+
| Chirpy   | oiseau  | 1998-09-11 |
| Whistler | oiseau  | 1997-12-09 |
| Claws    | chat    | 1994-03-17 |
| Fluffy   | chat    | 1993-02-04 |
| Fang     | chien   | 1990-08-27 |
| Bowser   | chien   | 1989-08-31 |
| Buffy    | chien   | 1989-05-13 |
| Puffball | hamster | 1999-03-30 |
| Slim     | serpent | 1996-04-29 |
+----------+---------+------------+
```

Notez que le mot-clè DESC est appliquè uniquement au nom de la colonne qui le prècéde (naissance) ; les valeurs espece continuent á être triées dans l'ordre croissant.

3.3.4.5 Calcul sur les Dates

MySQL fournit plusieurs fonctions que vous pouvez utiliser pour effectuer des calculs sur les dates, par exemple, pour calculer l'âge ou pour extraire des parties de date.

Pour dèterminer quel âge a chacun de vos animaux, vous devez calculer la diffèrence entre l'année en cours et l'année de naissance, puis soustraire á la date courante si la date du jour se produit plus tôt dans l'année civile que la date de naissance. La requête suivante montre, pour chaque animal, la date de naissance, la date courante, ainsi que l'âge en années.

```
mysql> SELECT nom, naissance, CURRENT_DATE,
    -> (YEAR(CURRENT_DATE)-YEAR(naissance))
    -> - (RIGHT(CURRENT_DATE,5)<RIGHT(naissance,5))
    -> AS age
    -> FROM animaux;
+----------+------------+--------------+------+
| nom      |naissance   | CURRENT_DATE | age  |
+----------+------------+--------------+------+
| Fluffy   | 1993-02-04 | 2001-08-29   |    8 |
| Claws    | 1994-03-17 | 2001-08-29   |    7 |
| Buffy    | 1989-05-13 | 2001-08-29   |   12 |
| Fang     | 1990-08-27 | 2001-08-29   |   11 |
| Bowser   | 1989-08-31 | 2001-08-29   |   11 |
| Chirpy   | 1998-09-11 | 2001-08-29   |    2 |
| Whistler | 1997-12-09 | 2001-08-29   |    3 |
| Slim     | 1996-04-29 | 2001-08-29   |    5 |
| Puffball | 1999-03-30 | 2001-08-29   |    2 |
+----------+------------+--------------+------+
```

Ici, YEAR() extrait l'année de la date et RIGHT() extrait les 5 caractéres les plus á droite de la date qui représentent MM-DD (année civile). La partie de l'expression qui compare les valeurs de MM-DD èvalue á 1 ou á 0, qui ajustent la diffèrence d'année á la baisse, si CURRENT_DATE se produit plus au dèbut de l'année que la naissance. L'expression compléte est un peu plus fine en utilisant un alias (age) pour produire un nom de colonne un peu plus significatif.

La requête fonctionne, mais le rèsultat pourrait être lu plus facilement si les lignes ètaient prèsentèes dans le même ordre. Cela peut être obtenu en ajoutant une clause ORDER BY nom pour trier le rèsultat par nom :

```
mysql> SELECT nom, naissance, CURRENT_DATE,
    -> (YEAR(CURRENT_DATE)-YEAR(naissance))
    -> - (RIGHT(CURRENT_DATE,5)<RIGHT(naissance,5))
    -> AS age
    -> FROM animaux ORDER BY nom;
+----------+------------+--------------+------+
| nom      |naissance   | CURRENT_DATE | age  |
```

```
+----------+------------+--------------+------+
| Bowser   | 1989-08-31 | 2001-08-29   |   11 |
| Buffy    | 1989-05-13 | 2001-08-29   |   12 |
| Chirpy   | 1998-09-11 | 2001-08-29   |    2 |
| Claws    | 1994-03-17 | 2001-08-29   |    7 |
| Fang     | 1990-08-27 | 2001-08-29   |   11 |
| Fluffy   | 1993-02-04 | 2001-08-29   |    8 |
| Puffball | 1999-03-30 | 2001-08-29   |    2 |
| Slim     | 1996-04-29 | 2001-08-29   |    5 |
| Whistler | 1997-12-09 | 2001-08-29   |    3 |
+----------+------------+--------------+------+
```

Pour trier le rèsultat par l'`age` plutôt que par le `nom`, utilisez simplement une clause `ORDER BY` diffèrente :

```
mysql> SELECT nom, naissance, CURRENT_DATE,
    -> (YEAR(CURRENT_DATE)-YEAR(naissance))
    -> - (RIGHT(CURRENT_DATE,5)<RIGHT(naissance,5))
    -> AS age
    -> FROM animaux ORDER BY age;
+----------+------------+--------------+------+
| nom      |naissance   | CURRENT_DATE | age  |
+----------+------------+--------------+------+
| Chirpy   | 1998-09-11 | 2001-08-29   |    2 |
| Puffball | 1999-03-30 | 2001-08-29   |    2 |
| Whistler | 1997-12-09 | 2001-08-29   |    3 |
| Slim     | 1996-04-29 | 2001-08-29   |    5 |
| Claws    | 1994-03-17 | 2001-08-29   |    7 |
| Fluffy   | 1993-02-04 | 2001-08-29   |    8 |
| Fang     | 1990-08-27 | 2001-08-29   |   11 |
| Bowser   | 1989-08-31 | 2001-08-29   |   11 |
| Buffy    | 1989-05-13 | 2001-08-29   |   12 |
+----------+------------+--------------+------+
```

Une requête similaire peut être utilisèe pour dèterminer l'âge qu'avait un animal á sa mort. Vous determinez les animaux qui le sont en regardant les valeurs `mort` qui ne valent pas `NULL`. Alors, pour ceux dont la valeur est non `NULL`, calculez la diffèrence entre la `mort` et la `naissance` :

```
mysql> SELECT nom, naissance, mort,
    -> (YEAR(death)-YEAR(naissance)) - (RIGHT(mort,5)<RIGHT(naissance,5))
    -> AS age
    -> FROM animaux WHERE mort IS NOT NULL ORDER BY age;
+--------+------------+------------+------+
| nom    |naissance   | mort       | age  |
+--------+------------+------------+------+
| Bowser | 1989-08-31 | 1995-07-29 |    5 |
+--------+------------+------------+------+
```

Cette requête utilise `mort IS NOT NULL` plutôt que `mort <> NULL` parce que `NULL` est une valeur spèciale. Cela sera expliquè plus tard. Voir Section 3.3.4.6 [Working with NULL], page 170.

Vous dèsirez savoir quels sont les animaux qui ont leur anniversaire le mois prochain ? Pour effectuer ce type de calculs, l'année et le jour ne sont pas utiles ; vous voulez simplement extraire le mois de la colonne `naissance`. MySQL fournit plusieurs fonctions d'extraction de parties de dates, comme `YEAR()`, `MONTH()`, et `DAYOFMONTH()`. `MONTH()` est la fonction appropriè dans notre cas. Pour voir comment cette fonction travaille, exècutez une requête simple qui retourne l'`naissance` et le `MONTH(naissance)` :

```
mysql> SELECT nom, naissance, MONTH(naissance) FROM animaux;
+----------+------------+------------------+
| nom      | naissance  | MONTH(naissance) |
+----------+------------+------------------+
| Fluffy   | 1993-02-04 |                2 |
| Claws    | 1994-03-17 |                3 |
| Buffy    | 1989-05-13 |                5 |
| Fang     | 1990-08-27 |                8 |
| Bowser   | 1989-08-31 |                8 |
| Chirpy   | 1998-09-11 |                9 |
| Whistler | 1997-12-09 |               12 |
| Slim     | 1996-04-29 |                4 |
| Puffball | 1999-03-30 |                3 |
+----------+------------+------------------+
```

Trouver les animaux qui ont leur anniversaire dans le mois suivant est aisè. Supposez que le mois courant est Avril. Donc, la valeur du mois est `4` et vous cherchez les animaux nès en Mai (mois 5) comme ceci :

```
mysql> SELECT nom, naissance FROM animaux WHERE MONTH(naissance) = 5;
+-------+------------+
| nom   |naissance   |
+-------+------------+
| Buffy | 1989-05-13 |
+-------+------------+
```

Il y a une petite complication si le mois courant est Dècembre, bien sûr. Vous ne pouvez pas uniquement ajouter 1 au numèro du mois courant (`12`) et chercher les animaux qui sont nès le mois numèro 13, parce qu'il n'existe pas. A la place, vous cherchez les animaux nès en Janvier (mois numèro 1).

Vous pouvez toujours ècrire une requête qui fonctionne quelque soit le mois courant. Comme cela, vous n'avez pas á utiliser un numèro de mois particulier dans votre requête. `DATE_ADD()` vous permet d'ajouter un intervalle de temps á une date donnèe. Si vous ajoutez un mois á la valeur de `NOW()`, et que vous extrayez le mois á l'aide de `MONTH()`, le rèsultat produit le mois dans lequel vous devez chercher un anniversaire :

```
mysql> SELECT nom, naissance FROM animaux
    -> WHERE MONTH(naissance) = MONTH(DATE_ADD(NOW(), INTERVAL 1 MONTH));
```

Une maniére diffèrente d'arriver au même rèsultat est d'ajouter `1` pour trouver le mois prochain aprés le mois courant (aprés l'usage de la fonction (`MOD`) pour ajouter á la valeur du mois la valeur `0` si il est de `12`) :

```
mysql> SELECT nom, naissance FROM animaux
    -> WHERE MONTH(naissance) = MOD(MONTH(NOW()), 12) + 1;
```

Notez que `MONTH` retourne un nombre entre 1 et 12. `MOD(quelquechose,12)` retourne un nombre entre 0 et 11. Donc, l'addition doit être faite aprés l'utilisation de la fonction `MOD()`, sinon, nous aurions un intervalle entre Novembre (11) et Janvier (1).

3.3.4.6 Travailler avec la valeur `NULL`

La valeur `NULL` peut être surprenante jusqu'á ce que vous vous y habituiez. Conceptuellement, `NULL` reprèsente une valeur qui manque, ou une valeur inconnue, et elle est traitèe diffèremment des autres valeurs. Pour tester la prèsence de la valeur `NULL`, vous ne pouvez pas utiliser les opèrateurs arithmètiques habituels comme `=`, `<`, ou `<>`. Pour le voir, il suffit d'essayer ceci :

```
mysql> SELECT 1 = NULL, 1 <> NULL, 1 < NULL, 1 > NULL;
+----------+-----------+----------+----------+
| 1 = NULL | 1 <> NULL | 1 < NULL | 1 > NULL |
+----------+-----------+----------+----------+
|     NULL |      NULL |     NULL |     NULL |
+----------+-----------+----------+----------+
```

Clairement, vous n'obtiendrez aucun rèsultat valable pour ces comparaisons. Utilisez les opèrateurs `IS NULL` et `IS NOT NULL` á la place :

```
mysql> SELECT 1 IS NULL, 1 IS NOT NULL;
+-----------+---------------+
| 1 IS NULL | 1 IS NOT NULL |
+-----------+---------------+
|         0 |             1 |
+-----------+---------------+
```

Notez que deux `NULL` sont considèrès comme ègaux lors que vous utilisez la clause `GROUP BY`.

Avec MySQL, 0 et `NULL` reprèsentent le boolèen faux, et tout le reste reprèsente le boolèen vrai. La valeur par dèfaut du boolèen vrai issue d'une comparaison est 1.

Lorsque vous utilisez la clause `ORDER BY`, les valeurs `NULL` sont toujours triées en premier, même si vous utilisez l'attribut `DESC`.

Ce traitement particulier de `NULL` explique pourquoi, dans la section prècèdente, il ètait nècessaire de dèterminer quel animal ne vivait plus en utilisant la fonction `mort IS NOT NULL` au lieu de `mort <> NULL`.

3.3.4.7 Recherche de modéles

MySQL fournit le standard SQL des recherches de modéles, basè sur une extension des expressions règuliéres similaires á celles utilisèes par les utilitaires Unix comme `vi`, `grep`, et `sed`.

La recherche de modéles SQL vous permet d'utiliser le caractére '_' pour trouver n'importe quel caractére et le caractére '%' pour trouver un nombre arbitraire de caractéres (y compris aucun caractére). Dans MySQL, la recherche de modéles est sensible á la casse par dèfaut. Quelques exemples vous sont prèsentès ici. Notez que vous n'utilisez ni = ni <> lorsque vous utilisez la recherche de modéles SQL ; utilisez les opèrateurs de comparaison LIKE ou NOT LIKE á la place.

Pour trouver les noms commençant par la lettre 'b' :

```
mysql> SELECT * FROM animaux WHERE nom LIKE "b%";
+--------+--------+--------+------+------------+------------+
| nom    | maitre | espece | sexe | naissance  | mort       |
+--------+--------+--------+------+------------+------------+
| Buffy  | Harold | chien  | f    | 1989-05-13 | NULL       |
| Bowser | Diane  | chien  | m    | 1989-08-31 | 1995-07-29 |
+--------+--------+--------+------+------------+------------+
```

Pour trouver les noms finissant par 'fy' :

```
mysql> SELECT * FROM animaux WHERE nom LIKE "%fy";
+--------+--------+---------+------+------------+-------+
| nom    | maitre | espece  | sexe | naissance  | mort  |
+--------+--------+---------+------+------------+-------+
| Fluffy | Harold | chat    | f    | 1993-02-04 | NULL  |
| Buffy  | Harold | chien   | f    | 1989-05-13 | NULL  |
+--------+--------+---------+------+------------+-------+
```

Pour trouver les noms contenant le caractéres 'w' :

```
mysql> SELECT * FROM animaux WHERE nom LIKE "%w%";
+----------+--------+---------+------+------------+------------+
| nom      | maitre | espece  | sexe | naissance  | mort       |
+----------+--------+---------+------+------------+------------+
| Claws    | Gwen   | chat    | m    | 1994-03-17 | NULL       |
| Bowser   | Diane  | chien   | m    | 1989-08-31 | 1995-07-29 |
| Whistler | Gwen   | oiseaux | NULL | 1997-12-09 | NULL       |
+----------+--------+---------+------+------------+------------+
```

Pour trouver les noms contenant exactement 5 caractéres, utilisez le caractére de recherche '_' :

```
mysql> SELECT * FROM animaux WHERE nom LIKE "_____";
+-------+--------+--------+------+------------+------+
| nom   | maitre | espece | sexe | naissance  | mort |
+-------+--------+--------+------+------------+------+
| Claws | Gwen   | chat   | m    | 1994-03-17 | NULL |
| Buffy | Harold | chien  | f    | 1989-05-13 | NULL |
+-------+--------+--------+------+------------+------+
```

L'autre type de recherche de modéles fourni par MySQL utilise les expressions règuliéres ètendues. Lorsque vous testez une recherche avec ce type de modéle, utilisez les opèrateurs REGEXP et NOT REGEXP (ou RLIKE et NOT RLIKE qui sont des synonymes).

Quelques caractèristiques des expressions règuliéres ètendues sont :

- Le caractére '`.`' trouve n'importe quel caractére.
- Une classe de caractéres '`[...]`' trouve n'importe quel caractére contenu entre les crochets. Par exemple, la classe de caractéres '`[abc]`' trouve le caractére '`a`', '`b`', ou '`c`'. Pour dèfinir un intervalle de caractéres, utilisez un trait d'union. La classe de caractéres '`[a-z]`' trouvera n'importe quel caractére minuscule, tout comme la classe '`[0-9]`' trouvera n'importe quel nombre.
- Le caractére '`*`' trouvera aucune ou plus d'instances du caractére qui le prècéde. Par exemple, '`x*`' trouvera n'importe quel nombre de fois le caractére '`x`', '`[0-9]*`' trouvera n'importe quel nombre, et '`.*`' trouvera n'importe quel nombre de fois n'importe quel caractére.
- Le modéle est trouvè s'il se produit n'importe oú dans la valeur testèe. (Les modéles SQL ne sont trouvès que s'ils sont prèsents en valeur entiére.)
- Pour ancrer un modéle de sorte qu'il soit trouvè au dèbut ou á la fin de valeur testèe, utilisez '`^`' au dèbut ou bien '`$`' á la fin du modéle.

Pour dèmontrer comment les expressions règuliéres fonctionnent, les requêtes `LIKE` vues prècèdemment ont ètè rèècrites pour utiliser `REGEXP`.

Pour trouver les noms qui commencent par la lettre '`b`', utilisez '`^`' pour trouver le dèbut du nom :

```
mysql> SELECT * FROM animaux WHERE nom REGEXP "^b";
+--------+--------+--------+------+------------+------------+
| nom    | maitre | espece | sexe | naissance  | mort       |
+--------+--------+--------+------+------------+------------+
| Buffy  | Harold | chien  | f    | 1989-05-13 | NULL       |
| Bowser | Diane  | chien  | m    | 1989-08-31 | 1995-07-29 |
+--------+--------+--------+------+------------+------------+
```

Avant la version 3.23.4 de MySQL, `REGEXP` ètait sensible á la casse, et la requête prècèdente ne retournait aucune ligne. Pour trouver la lettre '`b`' minuscule ou majuscule, utilisez cette requête á la place :

```
mysql> SELECT * FROM animaux WHERE nom REGEXP "^[bB]";
```

Depuis MySQL 3.23.4, pour forcer `REGEXP` á être sensible á la casse, utilisez le mot-clè `BINARY` pour faire de la chaîne, une chaîne binaire. Cette requête trouvera uniquement la lettre minuscule '`b`' au dèbut du nom :

```
mysql> SELECT * FROM animaux WHERE nom REGEXP BINARY "^b";
```

Pour trouver les noms finissant par '`fy`', utilisez '`$`' pour trouver la fin du nom :

```
mysql> SELECT * FROM animaux WHERE nom REGEXP "fy$";
+---------+--------+--------+-------+------------+------+
| nom     | maitre | espece | sexe  | naissance  | mort |
+---------+--------+--------+-------+------------+------+
| Fluffy  | Harold | chat   | f     | 1993-02-04 | NULL |
| Buffy   | Harold | chien  | f     | 1989-05-13 | NULL |
+---------+--------+--------+-------+------------+------+
```

Pour trouver les noms contenant la lettre '`w`' minuscule ou majuscule, utilisez la requête suivante :

```
mysql> SELECT * FROM animaux WHERE nom REGEXP "w";
+----------+--------+---------+------+------------+------------+
| nom      | maitre | espece  | sexe | naissance  | mort       |
+----------+--------+---------+------+------------+------------+
| Claws    | Gwen   | chat    | m    | 1994-03-17 | NULL       |
| Bowser   | Diane  | chien   | m    | 1989-08-31 | 1995-07-29 |
| Whistler | Gwen   | oiseaux | NULL | 1997-12-09 | NULL       |
+----------+--------+---------+------+------------+------------+
```

Parce qu'une expression règuliére est trouvèe si le modéle se trouve n'importe oú dans la valeur, il n'est pas nècessaire dans la requête prècèdente de mettre un joker de chaque côtè du modéle recherchè pour trouver la valeur entiére comme cela aurait ètè le cas en utilisant les modéles de recherche SQL.

Pour trouver les noms contenant exactement 5 caractéres, utilisez '`^`' et '`$`' pour trouver le dèbut et la fin du nom, et 5 instances de '`.`' au milieu :

```
mysql> SELECT * FROM animaux WHERE nom REGEXP "^.....$";
+-------+--------+--------+------+------------+------+
| nom   | maitre | espece | sexe | naissance  | mort |
+-------+--------+--------+------+------------+------+
| Claws | Gwen   | chat   | m    | 1994-03-17 | NULL |
| Buffy | Harold | chien  | f    | 1989-05-13 | NULL |
+-------+--------+--------+------+------------+------+
```

Vous pouvez aussi ècrire la requête suivante en utilisant l'opèrateur '`{n}`' "rèpèter-`n`-fois" :

```
mysql> SELECT * FROM animaux WHERE nom REGEXP "^.{5}$";
+-------+--------+--------+------+------------+------+
| nom   | maitre | espece | sexe | naissance  | mort |
+-------+--------+--------+------+------------+------+
| Claws | Gwen   | chat   | m    | 1994-03-17 | NULL |
| Buffy | Harold | chien  | f    | 1989-05-13 | NULL |
+-------+--------+--------+------+------------+------+
```

3.3.4.8 Compter les lignes

Les bases de donnèes sont souvent employèes pour rèpondre á la question : "Combien de fois un certain type de donnèes se trouve dans la table ?" Par exemple, vous aimeriez savoir combien d'animaux vous avez, ou bien combien d'animaux chaque propriètaire posséde, ou encore savoir diffèrentes choses concernant vos animaux.

Savoir combien vous avez d'animaux revient á se poser la question : "Combien de lignes y a-t-il dans la table `animaux` ?" parcequ'il y a un enregistrement par animaux. La fonction `COUNT()` compte le nombre de rèsultats non `NULL`, donc, la requête pour compter les animaux ressemble á ceci :

```
mysql> SELECT COUNT(*) FROM animaux;
+----------+
| COUNT(*) |
+----------+
|        9 |
```

```
+----------+
```

Vous pouvez trouver ègalement les noms des propriètaires des animaux. Vous pouvez utiliser `COUNT()` si vous voulez trouver combien d'animaux posséde chaque propriètaire :

```
mysql> SELECT maitre, COUNT(*) FROM animaux GROUP BY maitre;
+--------+----------+
| maitre | COUNT(*) |
+--------+----------+
| Benny  |        2 |
| Diane  |        2 |
| Gwen   |        3 |
| Harold |        2 |
+--------+----------+
```

Notez l'utilisation de la clause `GROUP BY` pour grouper tous les enregistrements par propriètaire. Sans cela, vous auriez le message d'erreur suivant :

```
mysql> SELECT maitre, COUNT(maitre) FROM animaux;
ERROR 1140 at line 1: Mixing of GROUP columns (MIN(),MAX(),COUNT()...)
with no GROUP columns is illegal if there is no GROUP BY clause
```

`COUNT()` et `GROUP BY` sont utiles pour caractèriser vos donnèes de diverses façons. Les exemples suivants montrent diffèrentes maniéres pour obtenir des statistiques sur les animaux.

Nombre d'animaux par espéce :

```
mysql> SELECT espece, COUNT(*) FROM animaux GROUP BY espece;
+---------+----------+
| espece  | COUNT(*) |
+---------+----------+
| oiseau  |        2 |
| chat    |        2 |
| chien   |        3 |
| hamster |        1 |
| serpent |        1 |
+---------+----------+
```

Nombre d'animaux par sexe :

```
mysql> SELECT sexe, COUNT(*) FROM animaux GROUP BY sexe;
+------+----------+
| sexe | COUNT(*) |
+------+----------+
| NULL |        1 |
| f    |        4 |
| m    |        4 |
+------+----------+
```

(Dans ce rèsultat, `NULL` indique les sexes inconnus.)

Nombre d'animaux par espéce et sexe :

```
mysql> SELECT espece, sexe, COUNT(*) FROM animaux GROUP BY espece, sexe;
+---------+------+----------+
| espece  | sexe | COUNT(*) |
```

```
+---------+------+----------+
| oiseau  | NULL |        1 |
| oiseau  | f    |        1 |
| chat    | f    |        1 |
| chat    | m    |        1 |
| chien   | f    |        1 |
| chien   | m    |        2 |
| hamster | f    |        1 |
| serpent | m    |        1 |
+---------+------+----------+
```

Vous n'avez pas besoin de rechercher une table entiére quand vous employez `COUNT()`. Par exemple, la requête prècèdente, si vous voulez trouver uniquement les chiens et les chats, ressemble á cela :

```
mysql> SELECT espece, sexe, COUNT(*) FROM animaux
    -> WHERE espece = "chien" OR espece = "chat"
    -> GROUP BY espece, sexe;
+---------+------+----------+
| espece  | sexe | COUNT(*) |
+---------+------+----------+
| chat    | f    |        1 |
| chat    | m    |        1 |
| chien   | f    |        1 |
| chien   | m    |        2 |
+---------+------+----------+
```

Ou bien, si vous voulez trouver le nombre d'animaux par sexe, uniquement pour les animaux dont le sexe est connu :

```
mysql> SELECT espece, sexe, COUNT(*) FROM animaux
    -> WHERE sexe IS NOT NULL
    -> GROUP BY espece, sexe;
+---------+------+----------+
| espece  | sexe | COUNT(*) |
+---------+------+----------+
| oiseau  | f    |        1 |
| chat    | f    |        1 |
| chat    | m    |        1 |
| chien   | f    |        1 |
| chien   | m    |        2 |
| hamster | f    |        1 |
| serpent | m    |        1 |
+---------+------+----------+
```

3.3.4.9 Utiliser plus d'une table

La table `animal` garde les enregistrements de vos animaux. Si vous voulez enregistrer d'autres informations concernant vos animaux, comme les èvènements de leurs vies, les

visites chez le vètèrinaire, ou encore lorsqu'ils ont mis bas, vous avez besoin d'une autre table. De quoi a besoin cette table ? Elle doit :

- Contenir le nom de l'animal pour savoir á quel animal cet èvènement se rattache.
- Une date pour savoir quand a eu lieu l'èvènement.
- Un champ qui dècrit l'èvènement.
- Un champ de type èvènement, si vous voulez être capable de cataloguer les èvènements.

En prenant cela en considèration, le code `CREATE TABLE` pour la table `evenement` doit ressembler á ceci :

```
mysql> CREATE TABLE evenement (nom VARCHAR(20), date DATE,
    -> type VARCHAR(15), remarque VARCHAR(255));
```

Tout comme la table `animal`, il est facile d'enregistrer les enregistrements initiaux en crèant un fichier texte dèlimitè par des tabulations, contenant l'information :

nom	date	type	remarque
Fluffy	1995-05-15	mise bas	4 chattons, 3 femelles, 1 mâles
Buffy	1993-06-23	mise bas	5 chiots, 2 femelles, 3 mâles
Buffy	1994-06-19	mise bas	3 chiots, 3 femelles
Chirpy	1999-03-21	vètèrinaire	needed beak straightened
Slim	1997-08-03	vètèrinaire	broken rib
Bowser	1991-10-12	chenil	
Fang	1991-10-12	chenil	
Fang	1998-08-28	anniversaire	Don d'un nouvel objet de mastication
Claws	1998-03-17	anniversaire	Don d'un nouveau collier anti-puces
Whistler	1998-12-09	anniversaire	Premier anniversaire

Chargez ces enregistrements comme cela :

```
mysql> LOAD DATA LOCAL INFILE "evenement.txt" INTO TABLE evenement;
```

En se basant sur ce que vous avez appris des requêtes effectuèes sur la table `animal` , vous devriez être capable de faire des recherches sur les enregistrements de la table `evenement` ; le principe est le même. Quand devez-vous vous demander si la table `evenement` est seule suffisante pour rèpondre á votre question ?

Supposez que vous voulez trouver l'âge de chaque animal lorsqu'il a mis bas. La table `evenement` indique quand cela s'est produit, mais pour le calcul de l'âge de la mére, vous avez besoin de sa date de naissance. Parce que ces informations sont stockèes dans la table `animal`, vous avez besoin des deux tables pour cette requête :

```
mysql> SELECT animal.nom,
    -> (TO_DAYS(date) - TO_DAYS(naissance))/365 AS age,
    -> remarque
    -> FROM animal, evenement
    -> WHERE animal.nom = evenement.nom AND type = "mise bas";
+--------+------+-------------------------------+
| nom    | age  | remarque                      |
```

```
+--------+------+--------------------------------+
| Fluffy | 2.27 | 4 chattons, 3 femelles, 1 mâle |
| Buffy  | 4.12 | 5 chiots, 2 femelles, 3 mâles  |
| Buffy  | 5.10 | 3 chiots, 3 femelles           |
+--------+------+--------------------------------+
```

Il y a plusieurs choses á noter concernant cette requête :

- La clause `FROM` liste les deux tables parce que la requête a besoin d'informations contenues dans ces deux tables.
- Lorsque vous combinez (joignez) des informations provenant de plusieurs tables, vous devez spècifier quels enregistrements d'une table peuvent être associès á quels enregistrements des autres tables. C'est aisè parce qu'elles ont toutes les deux une colonne `nom`. La requête utilise la clause `WHERE` pour faire correspondre les enregistrements des deux tables sur les valeurs de la colonne `nom`.
- Parce que la colonne `nom` apparaît dans les deux tables, vous devez être explicite concernant la table que vous utilisez lorsque vous vous rèfèrez á cette colonne. C'est fait en faisant prècèder le nom de la colonne par le nom de la table.

Vous n'avez pas besoin de deux tables diffèrentes pour effectuer une jointure. Quelques fois, c'est plus facile de joindre une table sur elle-même, si vous voulez comparer des enregistrements dans une table avec d'autres enregistrements de la même table. Par exemple, pour trouver des paires multiples parmi vos animaux, vous pouvez joindre la table `animal` sur elle-même pour trouver les paires mâles / femelles par rapport á l'espéce :

```
mysql> SELECT p1.nom, p1.sexe, p2.nom, p2.sexe, p1.espece
    -> FROM animal AS p1, animal AS p2
    -> WHERE p1.espece = p2.espece AND p1.sexe = "f" AND p2.sexe = "m";
+--------+------+--------+------+---------+
| nom    | sexe | nom    | sexe | espece  |
+--------+------+--------+------+---------+
| Fluffy | f    | Claws  | m    | chat    |
| Buffy  | f    | Fang   | m    | chien   |
| Buffy  | f    | Bowser | m    | chien   |
+--------+------+--------+------+---------+
```

Dans cette requête, nous avons spècifiè des alias pour les noms de tables dans l'ordre de rèfèrence des colonnes et ainsi maintenir directement á quelle instance de la table chaque colonne est associèe.

3.4 Obtenir des informations á propos des bases de donnèes et des tables

Que faire si vous oubliez le nom d'une base de donnèes ou d'une table, ou bien encore la structure d'une table donnèe (par exemple, comment se nomment ses colonnes) ?

MySQL rèpond á ce probléme en fournissant plusieurs commandes qui renvoient des informations á propos des tables et des bases de donnèes les contenant.

Vous avez dèjá vu `SHOW DATABASES` qui liste les bases de donnèes gèrèes par le serveur. Pour trouver quelle base de donnèes est actuellement sèlectionnèe, utilisez la fonction `DATABASE()` :

```
mysql> SELECT DATABASE();
+------------+
| DATABASE() |
+------------+
| menagerie  |
+------------+
```

Si vous n'avez encore sèlectionnè aucune base de donnèes, le rèsultat est vide.

Pour trouver quelles sont les tables que la base contient (par exemple, quand vous n'êtes pas sûr du nom d'une table), utilisez cette commande :

```
mysql> SHOW TABLES;
+---------------------+
| Tables in menagerie |
+---------------------+
| evenement           |
| animal              |
+---------------------+
```

Si vous voulez en savoir d'avantage sur la structure d'une table, la commande `DESCRIBE` est utile ; elle fournit des informations sur chaque colonne de la table :

```
mysql> DESCRIBE animal;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| nom       | varchar(20) | YES  |     | NULL    |       |
| maitre    | varchar(20) | YES  |     | NULL    |       |
| espece    | varchar(20) | YES  |     | NULL    |       |
| sexe      | char(1)     | YES  |     | NULL    |       |
| naissance | date        | YES  |     | NULL    |       |
| mort      | date        | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
```

`Field` indique le nom de la colonne, `Type` est son type de donnèes, `NULL` indique si la colonne peut contenir des valeurs `NULL`, `Key` indique si la colonne est indexèe et `Default` spècifie la valeur par dèfaut de la colonne.

Si vous avez des index sur une table, `SHOW INDEX FROM nom_de_table` vous fournira des informations sur elles.

3.5 Exemples de requêtes usuelles

Voilá des exemples qui vous serviront á rèsoudre les problémes communs avec MySQL.

Certains exemples utilisent la table `shop` pour sauvegarder le prix de chaque article (numèro de l'èlèment) pour certains vendeurs (dealers). En supposant que chaque vendeur á un prix fixe pour chaque article, le couple (`article`, `dealer`) est une clef primaire pour les enregistrements.

Dèmarrez le client en ligne de commande `mysql` et sèlectionnez une base de donnèes :

```
mysql nom-base-données
```

(Dans la plupart des installations de MySQL, vous pouvez utiliser la base de données 'test').
Vous pouvez crèer la table d'exemple de la façon suivante :

```
CREATE TABLE shop (
 article INT(4) UNSIGNED ZEROFILL DEFAULT '0000' NOT NULL,
 dealer  CHAR(20)                 DEFAULT ''     NOT NULL,
 price   DOUBLE(16,2)             DEFAULT '0.00' NOT NULL,
 PRIMARY KEY(article, dealer));

INSERT INTO shop VALUES
(1,'A',3.45),(1,'B',3.99),(2,'A',10.99),(3,'B',1.45),(3,'C',1.69),
(3,'D',1.25),(4,'D',19.95);
```

Ok, les données d'exemple sont :

```
mysql> SELECT * FROM shop;

+---------+--------+-------+
| article | dealer | price |
+---------+--------+-------+
|    0001 | A      |  3.45 |
|    0001 | B      |  3.99 |
|    0002 | A      | 10.99 |
|    0003 | B      |  1.45 |
|    0003 | C      |  1.69 |
|    0003 | D      |  1.25 |
|    0004 | D      | 19.95 |
+---------+--------+-------+
```

3.5.1 La valeur maximale d'une colonne

"Quel est le numèro du plus grand èlèment ?"

```
SELECT MAX(article) AS article FROM shop

+---------+
| article |
+---------+
|       4 |
+---------+
```

3.5.2 La ligne contenant le maximum d'une certaine colonne

"Trouvez le numèro, vendeur et prix de l'article le plus cher."
En ANSI SQL cela est facilement fait avec une sous-requête :

```
SELECT article, dealer, price
FROM   shop
WHERE  price=(SELECT MAX(price) FROM shop)
```

En MySQL (qui ne gére pas encore les sous-requêtes), vous devez le faire en deux temps :

1. Obtenir le plus grand prix de la table avec une requête `SELECT`.
2. Utiliser cette valeur avec cette requête :

```
SELECT article, dealer, price
FROM   shop
WHERE  price=19.95
```

Une autre solution est de trier toutes les lignes en ordre dècroissant des prix et ne choisir que la premiére ligne avec la clause `LIMIT` qui est spècifique á MySQL :

```
SELECT article, dealer, price
FROM   shop
ORDER BY price DESC
LIMIT 1
```

NOTE : s'il y'a beaucoup d'articles chers (par exemple, chaque 19.95) la solution avec `LIMIT` n'en montre qu'un !.

3.5.3 Maximum d'une colonne par groupe

"Quel est le plus grand prix par article ?"

```
SELECT article, MAX(price) AS price
FROM   shop
GROUP BY article

+---------+-------+
| article | price |
+---------+-------+
|    0001 |  3.99 |
|    0002 | 10.99 |
|    0003 |  1.69 |
|    0004 | 19.95 |
+---------+-------+
```

3.5.4 La ligne contenant la plus grande valeur d'un certain champ par rapport á un groupe

"Pour chaque article, trouvez le ou les vendeurs avec le plus haut prix."

En ANSI SQL, je l'aurais fait de cette façon avec une sous-requête :

```
SELECT article, dealer, price
FROM   shop s1
WHERE  price=(SELECT MAX(s2.price)
              FROM shop s2
              WHERE s1.article = s2.article);
```

En MySQL il vaut mieux le faire en plusieurs ètapes :

1. Rècupèrer la liste de (article, plusgrandprix).
2. Pour chaque article, rècupèrer la ligne qui a le plus grand prix stockè.

Cela se fait facilement avec une table temporaire :

```
CREATE TEMPORARY TABLE tmp (
        article INT(4) UNSIGNED ZEROFILL DEFAULT '0000' NOT NULL,
        price   DOUBLE(16,2)             DEFAULT '0.00' NOT NULL);

LOCK TABLES shop read;

INSERT INTO tmp SELECT article, MAX(price) FROM shop GROUP BY article;

SELECT shop.article, dealer, shop.price FROM shop, tmp
WHERE shop.article=tmp.article AND shop.price=tmp.price;

UNLOCK TABLES;

DROP TABLE tmp;
```

Si vous n'utilisez pas une table `TEMPORARY`, vous devez aussi verrouiller celle-ci.

"Peut-on le faire avec une seule requête ?"

Oui, mais en utilisant une astuce inefficace que j'appelle "astuce du MAX-CONCAT":

```
SELECT article,
       SUBSTRING( MAX( CONCAT(LPAD(price,6,'0'),dealer) ), 7) AS dealer,
  0.00+LEFT(      MAX( CONCAT(LPAD(price,6,'0'),dealer) ), 6) AS price
FROM   shop
GROUP BY article;

+---------+--------+-------+
| article | dealer | price |
+---------+--------+-------+
|    0001 | B      |  3.99 |
|    0002 | A      | 10.99 |
|    0003 | C      |  1.69 |
|    0004 | D      | 19.95 |
+---------+--------+-------+
```

Le dernier exemple peut, bien sûr, être amèliorè en dècoupant les colonnes concatènèes dans le client.

3.5.5 Utiliser les variables utilisateur

Vous pouvez utiliser les variables utilisateur de MySQL pour garder des rèsultats en mèmoire sans avoir á les enregistrer dans des variables temporaires du client. Voir Section 6.1.4 [Variables], page 409.

Par exemple, pour trouver l'article avec le plus haut et le plus bas prix, vous pouvez faire :

```
mysql> SELECT @min_price:=MIN(price),@max_price:=MAX(price) FROM shop;
mysql> SELECT * FROM shop WHERE price=@min_price OR price=@max_price;
+---------+--------+-------+
| article | dealer | price |
```

```
+---------+--------+-------+
|    0003 | D      |  1.25 |
|    0004 | D      | 19.95 |
+---------+--------+-------+
```

3.5.6 Utiliser les clefs ètrangéres

Depuis la version 3.23.44 de MySQL, les tables `InnoDB` supportent les contraintes des clefs ètrangéres. Voir Section 7.5 [InnoDB], page 544. Consultez aussi Section 1.7.4.5 [ANSI diff Foreign Keys], page 41.

Actuellement, vous n'avez pas besoin de clefs ètrangéres pour rèaliser des jointures entre les tables. La seule chose que MySQL ne fait pas encore (avec les types autres que `InnoDB`), est `CHECK` pour s'assurer que que la clef que vous utilisez existe bien dans la ou les tables que vous rèfèrencez et il n'efface pas automatiquement les lignes d'une table avec une dèfinition de clef ètrangére. Si vous utilisez vos clefs comme une clef normale, tout marchera parfaitement :

```
CREATE TABLE person (
    id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    name CHAR(60) NOT NULL,
    PRIMARY KEY (id)
);

CREATE TABLE shirt (
    id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    style ENUM('t-shirt', 'polo', 'dress') NOT NULL,
    color ENUM('red', 'blue', 'orange', 'white', 'black') NOT NULL,
    owner SMALLINT UNSIGNED NOT NULL REFERENCES person(id),
    PRIMARY KEY (id)
);


INSERT INTO person VALUES (NULL, 'Antonio Paz');

INSERT INTO shirt VALUES
(NULL, 'polo', 'blue', LAST_INSERT_ID()),
(NULL, 'dress', 'white', LAST_INSERT_ID()),
(NULL, 't-shirt', 'blue', LAST_INSERT_ID());


INSERT INTO person VALUES (NULL, 'Lilliana Angelovska');

INSERT INTO shirt VALUES
(NULL, 'dress', 'orange', LAST_INSERT_ID()),
(NULL, 'polo', 'red', LAST_INSERT_ID()),
(NULL, 'dress', 'blue', LAST_INSERT_ID()),
(NULL, 't-shirt', 'white', LAST_INSERT_ID());
```

```
SELECT * FROM person;
+----+--------------------+
| id | name               |
+----+--------------------+
|  1 | Antonio Paz        |
|  2 | Lilliana Angelovska |
+----+--------------------+

SELECT * FROM shirt;
+----+---------+--------+-------+
| id | style   | color  | owner |
+----+---------+--------+-------+
|  1 | polo    | blue   |     1 |
|  2 | dress   | white  |     1 |
|  3 | t-shirt | blue   |     1 |
|  4 | dress   | orange |     2 |
|  5 | polo    | red    |     2 |
|  6 | dress   | blue   |     2 |
|  7 | t-shirt | white  |     2 |
+----+---------+--------+-------+


SELECT s.* FROM person p, shirt s
 WHERE p.name LIKE 'Lilliana%'
   AND s.owner = p.id
   AND s.color <> 'white';

+----+-------+--------+-------+
| id | style | color  | owner |
+----+-------+--------+-------+
|  4 | dress | orange |     2 |
|  5 | polo  | red    |     2 |
|  6 | dress | blue   |     2 |
+----+-------+--------+-------+
```

3.5.7 Recherche sur deux clefs

MySQL n'optimise pas encore quand vous effectuez des recherches sur deux clefs diffèrentes combinèes avec `OR` (la recherche sur une clef avec diffèrentes parties `OR` est elle pas mal optimisèe) :

```
SELECT champ1_index, champ2_index FROM test_table WHERE champ1_index = '1'
OR  champ2_index = '1'
```

La raison est que nous n'avons pas trouvè le temps suffisant pour parvenir á un moyen efficace de gèrer cela dans un cas gènèral. (En comparaison, la gestion de `AND` est maintenant complétement gènèrale et fonctionne trés bien.)

Pour le moment, vous pouvez rèsoudre ce probléme efficacement en utilisant une table temporaire (`TEMPORARY`). Ce type d'optimisation est trés utile si vous utilisez des requêtes trés complexes et que le serveur SQL fait une optimisation dans le mauvais ordre.

```
CREATE TEMPORARY TABLE tmp
SELECT champ1_index, champ2_index FROM test_table WHERE champ1_index = '1';
INSERT INTO tmp
SELECT champ1_index, champ2_index FROM test_table WHERE champ2_index = '1';
SELECT * from tmp;
DROP TABLE tmp;
```

La mèthode ci-dessus pour rèsoudre cette requête est en effet une `UNION` de deux requêtes. Voir Section 6.4.1.2 [UNION], page 487.

3.5.8 Calculer les visites par jour

Ce qui suit donne une idèe d'une utilisation des fonctions de bits pour calculer le nombre de jours par mois oú un utilisateur a visitè une page web.

```
CREATE TABLE t1 (annee YEAR(4), mois INT(2) UNSIGNED ZEROFILL,
             jour INT(2) UNSIGNED ZEROFILL);
INSERT INTO t1 VALUES(2000,1,1),(2000,1,20),(2000,1,30),(2000,2,2),
            (2000,2,23),(2000,2,23);
SELECT annee,mois,BIT_COUNT(BIT_OR(1<<jour)) AS jours FROM t1
       GROUP BY annee,mois;

Qui retourne :

+-------+-------+-------+
| annee | mois  | jours |
+-------+-------+-------+
|  2000 |    01 |     3 |
|  2000 |    02 |     2 |
+-------+-------+-------+
```

Ce qui prècéde calcule le nombre de jours diffèrents qui a ètè utilisè pour une combinaison annèe/mois, avec suppression automatique des doublons.

3.5.9 Utiliser `AUTO_INCREMENT`

L'attribut `AUTO_INCREMENT` peut être utilisè pour gènèrer un identifiant unique pour les nouvelles lignes :

```
CREATE TABLE animals (
             id MEDIUMINT NOT NULL AUTO_INCREMENT,
             name CHAR(30) NOT NULL,
             PRIMARY KEY (id)
```

```
            );
INSERT INTO animals (name) VALUES ("dog"),("cat"),("penguin"),
                                  ("lax"),("whale");
SELECT * FROM animals;

Qui retourne :

+----+---------+
| id | name    |
+----+---------+
|  1 | dog     |
|  2 | cat     |
|  3 | penguin |
|  4 | lax     |
|  5 | whale   |
+----+---------+
```

Vous pouvez obtenir la valeur utilisèe de la clef `AUTO_INCREMENT` avec la fonction SQL `LAST_INSERT_ID()` ou la fonction d'API `mysql_insert_id()`.

Note : Pour une insertion multi-lignes, `LAST_INSERT_ID()`/`mysql_insert_id()` retourneront la clef `AUTO_INCREMENT` de la **premiére** ligne insèrèe. Cela permet de reproduire les insertions multi-lignes sur d'autres services.

Pour les tables MyISAM et BDB vous pouvez spècifier `AUTO_INCREMENT` sur une colonne secondaire d'une clef multi-colonnes. Dans ce cas, la valeur gènèrèe pour la colonne auto-incrementèe est calculèe de la façon suivante : `MAX(auto_increment_column)+1) WHERE prefix=given-prefix`. C'est utile lorsque vous voulez placer des donnèes dans des groupes ordonnès.

```
CREATE TABLE animals (
             grp ENUM('fish','mammal','bird') NOT NULL,
             id MEDIUMINT NOT NULL AUTO_INCREMENT,
             name CHAR(30) NOT NULL,
             PRIMARY KEY (grp,id)
             );
INSERT INTO animals (grp,name) VALUES("mammal","dog"),("mammal","cat"),
                  ("bird","penguin"),("fish","lax"),("mammal","whale");
SELECT * FROM animals ORDER BY grp,id;

Qui retourne :

+--------+----+---------+
| grp    | id | name    |
+--------+----+---------+
| fish   |  1 | lax     |
| mammal |  1 | dog     |
| mammal |  2 | cat     |
| mammal |  3 | whale   |
| bird   |  1 | penguin |
```

```
+--------+----+---------+
```

Notez que dans ce cas, la valeur d'`AUTO_INCREMENT` sera rèutilisèe si vous effacez la ligne avec la plus grande valeur d'`AUTO_INCREMENT` tous groupes confondus.

3.6 Utilisation de `mysql` en mode batch

Dans les sections prècèdentes, vous avez utilisè `mysql` inter activement pour entrer vos requêtes et voir les rèsultats. Vous pouvez aussi utiliser `mysql` en mode batch. Pour ce faire, placez les commandes que vous voulez exècuter dans un fichier, puis dites á `mysql` de lire les entrèes á partir de celui-ci :

```
shell> mysql < fichier-batch
```

Si vous utilisez `mysql` sous windows et que vous avez des caractéres spèciaux dans le fichier qui posent problémes, vous pouvez faire :

```
dos> mysql -e "source fichier-batch"
```

Si vous devez spècifier les paramétres de connexion en ligne de commande, la commande ressemblera á ca :

```
shell> mysql -h hôte -u utilisateur -p < fichier-batch
Enter password: ********
```

Lorsque vous utilisez `mysql` de cette façon, vous crèez un fichier de script, puis vous l'exècutez.

Si vous voulez que le script continue, même si il y'a des erreurs, vous devez utiliser l'option `--force` de la ligne de commande.

Pourquoi utilisez un script ? Voici quelques raisons :

- Si vous utilisez une requête de façon rèpètitive (c'est á dire, chaque jour, ou chaque semaine), en faire un script vous èvitera de la rèècrire chaque fois.
- Vous pouvez gènèrer de nouvelles requêtes á partir de requêtes existantes et similaires en copiant et èditant des fichiers de scripts.
- Ce mode peut aussi être utile lors du dèveloppement d'une requête, particuliérement pour les commandes sur plusieurs lignes ou plusieurs sèquences de commandes. Si vous commetez une erreur, vous n'avez pas á tout rècrire. Editez juste votre script pour corriger l'erreur et dites á `mysql` de l'exècuter á nouveau.
- Si vous avez une requête qui produit beaucoup d'affichage, vous pouvez le rediriger vers un visionneur plutôt que de le regarder dèfiler sur votre ècran :

  ```
  shell> mysql < fichier-batch | more
  ```

- Vous pouvez capturer l'affichage dans un fichier pour un traitement ultèrieur :

  ```
  shell> mysql < fichier_batch > mysql.out
  ```

- Vous pouvez distribuer votre script á d'autres personnes pour qu'elles l'exècutent.
- Quelques situations ne permettent pas une utilisation interactive, par exemple, quand vous exècutez une requête á partir d'une tâche `cron`. Dans ce cas, vous devez utiliser le mode batch.

Le format d'affichage par dèfaut est diffèrent (plus concis) lorsque vous exècutez `mysql` en mode batch de celui utilisè inter activement. Par exemple, le rèsultat de `SELECT DISTINCT espece FROM animal` ressemble á ça inter activement :

```
+---------+
| espece  |
+---------+
| oiseau  |
| chat    |
| chien   |
| hamster |
| serpent |
+---------+
```

Mais á ça en mode batch :

```
espece
oiseau
chat
chien
hamster
serpent
```

Si vous voulez le format d'affichage interactif en mode batch, utilisez `mysql -t`. Pour ècrire les commandes exècutez dans la sortie, utilisez `mysql -vvv`.

Vous pouvez aussi utiliser un script á partir de l'invite `mysql` en utilisant la commande `source` :

```
mysql> source nom_fichier;
```

3.7 Requêtes du projet Twin

A Analytikerna et Lentus, nous avons eu á mettre en place le partie systéme et base de donnèes d'un grand projet de recherche. Ce projet est une collaboration entre l'institut de mèdecine environnementale de l'institut de Karolinska Stockholm et la section de recherche clinique d'âge et de psychologie á l'universitè de la Californie du sud.

Le projet nècessite une partie de rècolte d'informations oú tous les jumeaux en Suéde de plus de 65 ans sont contactès par tèlèphone. Ceux qui rèpondent á certains critéres sont admis á la seconde ètape. Dans celle-ci, les jumeaux qui veulent participer rencontrent une èquipe de mèdecins/infirmiers. Les examens incluent des examens physiques et neuropsychologiques, des tests en laboratoire, de la neuro-imagerie, des ètudes psychologiques et de la collecte d'informations relatives á la famille. En plus de tout cela, les donnèes á propos des facteurs de risques mèdicaux et environnementaux sont collectèes.

Plus d'informations á propos de l'ètude Twin peuvent être trouvèes sur : `http://www.imm.ki.se/TWIN/TWINUKW.HTM`

La derniére partie de ce projet est administrèe avec une interface web ècrite en utilisant Perl et MySQL.

Chaque nuit, toutes les informations des interviews sont stockèes dans une base de donnèes MySQL.

3.7.1 Trouver tous les jumeaux rèpondant aux critéres

La requête suivante a ètè utilisèe pour dèterminer qui participerait á la seconde partie du projet :

```
SELECT
        CONCAT(p1.id, p1.tvab) + 0 AS tvid,
        CONCAT(p1.christian_name, " ", p1.surname) AS Name,
        p1.postal_code AS Code,
        p1.city AS City,
        pg.abrev AS Area,
        IF(td.participation = "Aborted", "A", " ") AS A,
        p1.dead AS dead1,
        l.event AS event1,
        td.suspect AS tsuspect1,
        id.suspect AS isuspect1,
        td.severe AS tsevere1,
        id.severe AS isevere1,
        p2.dead AS dead2,
        l2.event AS event2,
        h2.nurse AS nurse2,
        h2.doctor AS doctor2,
        td2.suspect AS tsuspect2,
        id2.suspect AS isuspect2,
        td2.severe AS tsevere2,
        id2.severe AS isevere2,
        l.finish_date
FROM
        twin_project AS tp
        /* For Twin 1 */
        LEFT JOIN twin_data AS td ON tp.id = td.id
                  AND tp.tvab = td.tvab
        LEFT JOIN informant_data AS id ON tp.id = id.id
                  AND tp.tvab = id.tvab
        LEFT JOIN harmony AS h ON tp.id = h.id
                  AND tp.tvab = h.tvab
        LEFT JOIN lentus AS l ON tp.id = l.id
                  AND tp.tvab = l.tvab
        /* For Twin 2 */
        LEFT JOIN twin_data AS td2 ON p2.id = td2.id
                  AND p2.tvab = td2.tvab
        LEFT JOIN informant_data AS id2 ON p2.id = id2.id
                  AND p2.tvab = id2.tvab
        LEFT JOIN harmony AS h2 ON p2.id = h2.id
                  AND p2.tvab = h2.tvab
        LEFT JOIN lentus AS l2 ON p2.id = l2.id
                  AND p2.tvab = l2.tvab,
```

```
        person_data AS p1,
        person_data AS p2,
        postal_groups AS pg
WHERE
        /* p1 gets main twin and p2 gets his/her twin. */
        /* ptvab is a field inverted from tvab */
        p1.id = tp.id AND p1.tvab = tp.tvab AND
        p2.id = p1.id AND p2.ptvab = p1.tvab AND
        /* Just the sceening survey */
        tp.survey_no = 5 AND
        /* Skip if partner died before 65 but allow emigration (dead=9) */
        (p2.dead = 0 OR p2.dead = 9 OR
         (p2.dead = 1 AND
          (p2.death_date = 0 OR
           (((TO_DAYS(p2.death_date) - TO_DAYS(p2.birthday)) / 365)
            >= 65))))
        AND
        (
        /* Twin is suspect */
        (td.future_contact = 'Yes' AND td.suspect = 2) OR
        /* Twin is suspect - Informant is Blessed */
        (td.future_contact = 'Yes' AND td.suspect = 1
                                   AND id.suspect = 1) OR
        /* No twin - Informant is Blessed */
        (ISNULL(td.suspect) AND id.suspect = 1
                            AND id.future_contact = 'Yes') OR
        /* Twin broken off - Informant is Blessed */
        (td.participation = 'Aborted'
         AND id.suspect = 1 AND id.future_contact = 'Yes') OR
        /* Twin broken off - No inform - Have partner */
        (td.participation = 'Aborted' AND ISNULL(id.suspect)
                                      AND p2.dead = 0))
        AND
        l.event = 'Finished'
        /* Get at area code */
        AND SUBSTRING(p1.postal_code, 1, 2) = pg.code
        /* Not already distributed */
        AND (h.nurse IS NULL OR h.nurse=00 OR h.doctor=00)
        /* Has not refused or been aborted */
        AND NOT (h.status = 'Refused' OR h.status = 'Aborted'
        OR h.status = 'Died' OR h.status = 'Other')
ORDER BY
        tvid;
```

Quelques explications :

`CONCAT(p1.id, p1.tvab) + 0 AS tvid`
: Nous voulons trier la concatènation de `id` et `tvab` dans un ordre numèrique. Ajouter `0` au rèsultat force MySQL á le considèrer comme un nombre.

colonne `id`
: Identifie une paire de jumeaux. C'est un clef dans toutes les tables.

colonne `tvab`
: Identifie un jumeau dans une paire. Valeur `1` ou `2`.

colonne `ptvab`
: Inverse de `tvab`. Si `tvab` est `1` c'est ègal á `2`, et vice-versa. Elle existe pour diminuer la frappe et faciliter la tâche á MySQL lors de l'optimisation de la requête.

Cette requête montre, entre autres, comment faire pour consulter une table depuis cette même table en utilisant une jointure (`p1` et `p2`). Dans cet exemple, est utilisè pour chercher quel partenaire du projet est dècèdè avant l'âge de 65 ans. Si c'est le cas, la ligne n'est pas retournèe.

Tout ce qui prècéde existe dans toutes les tables avec des informations relatives aux jumeaux. Nous avons une clè sur les champs `id,tvab` (toutes les tables), et sur les champs `id,ptvab` (`person_data`) pour accèlèrer les requêtes.

Sur notre machine de production (un 200MHz UltraSPARC), cette requête retourne prés de 150-200 lignes et prend moins d'une seconde.

Le nombre d'enregistrements dans les tables utilisèes plus haut :

Table	**Lignes**
`person_data`	71074
`lentus`	5291
`twin_project`	5286
`twin_data`	2012
`informant_data`	663
`harmony`	381
`postal_groups`	100

3.7.2 Afficher une table avec l'ètat des paires de jumeaux

Chaque entrevue se finit avec un code d'ètat, appelè `event`. La requête ci-dessous montre comment afficher une table avec toutes les paires, rassemblèes par code d'ètat. Elle indique combien de paires ont terminè, combien de paires ont á moitiè terminè et combien on refusè, etc.

```
SELECT
        t1.event,
        t2.event,
        COUNT(*)
FROM
        lentus AS t1,
        lentus AS t2,
        twin_project AS tp
```

```
WHERE
        /* We are looking at one pair at a time */
        t1.id = tp.id
        AND t1.tvab=tp.tvab
        AND t1.id = t2.id
        /* Just the sceening survey */
        AND tp.survey_no = 5
        /* This makes each pair only appear once */
        AND t1.tvab='1' AND t2.tvab='2'
GROUP BY
        t1.event, t2.event;
```

3.8 Utilisation de MySQL avec Apache

Il existe des programmes vous permettant d'identifier vos utilisateurs á l'aide d'une base MySQL et qui vous permettent aussi de crèer des journaux de log dans vos table MySQL. Voir Section 1.6.1 [Portals], page 23.

Vous pouvez changer le format d'archivage d'Apache pour le rendre plus facilement lisible par MySQL en mettant ce qui suit dans le fichier de configuration d'Apache :

```
LogFormat \
        "\"%h\",%{%Y%m%d%H%M%S}t,%>s,\"%b\",\"%{Content-Type}o\",  \
        \"%U\",\"%{Referer}i\",\"%{User-Agent}i\""
```

Avec MySQL, vous pouvez exècuter une requête de la sorte :

```
LOAD DATA INFILE '/local/access_log' INTO TABLE table_name
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' ESCAPED BY '\\'
```

4 Administration du serveur

4.1 Configuration de MySQL

4.1.1 Options de ligne de commande de `mysqld`

Dans la plupart des cas, vous devrez modifier les options de mysqld dans le fichier d'options. Voir Section 4.1.2 [Option files], page 198.

`mysqld` et `mysqld.server` lisent les options des groupes `mysqld` et `server`. `mysqld_safe` lit les options des groupes `mysqld`, `server`, `mysqld_safe` et `safe_mysqld`. Un serveur MySQL intègrè lit gènèralement les options dans les groupes `server`, `embedded` et `xxxxx_SERVER`, oú `xxxxx` est le nom de l'application.

`mysqld` accepte les options de ligne de commande suivantes :

`--ansi` Utilise la syntaxe ANSI SQL au lieu de la syntaxe MySQL. Voir Section 1.7.2 [ANSI mode], page 33.

`-b, --basedir=path`
Chemin jusqu'au dossier d'installation. Tous les chemins sont gènèralement relatifs á celui-ci.

`--big-tables`
Autorise la sauvegarde de grands rèsultats dans des fichiers temporaires. Cela rèsout le probléme des erreurs 'table full', mais ralentit les requêtes alors que des tables en mèmoire suffirait. Depuis la version version 3.23.2, MySQL est capable de rèsoudre automatiquement ce probléme en utilisant de la mèmoire pour toutes les tables temporaires de petite taille, et en passant sur le disque au besoin.

`--bind-address=IP`
L'adresse IP á utiliser.

`--character-sets-dir=path`
Dossier contenant les jeux de caractéres. Voir Section 4.6.1 [Character sets], page 286.

`--chroot=path`
Met le dèmon `mysqld` en environnement chroot au dèmarrage. Recommandè pour la sècuritè. Cela limite les commandes `LOAD DATA INFILE` et `SELECT ... INTO OUTFILE`.

`--core-file`
Ecrire le fichier core lorsque `mysqld` s'arrête inopinèement. Pour certains fichiers, vous devez aussi spècifier `--core-file-size` á `safe_mysqld`. Voir Section 4.7.2 [`safe_mysqld`], page 292. Notez que sur certains systémes, comme Solaris, vous n'aurez pas de fichier de core si vous avez aussi utilisè l'option `--user`.

`-h, --datadir=path`
: Chemin jusqu'au dossier des bases.

`--debug[...]=`
: Si MySQL est configurè avec `--with-debug`, vous pouvez utiliser cette option pour obtenir un fichier de trace de ce que `mysqld` fait. Voir Section E.1.2 [Making trace files], page 816.

`--default-character-set=charset`
: Spècifie le jeu de caractéres par dèfaut. Voir Section 4.6.1 [Character sets], page 286.

`--default-table-type=type`
: Spècifie le type de table par dèfaut. Voir Chapitre 7 [Table types], page 531.

`--delay-key-write[= OFF | ON | ALL]`
: Comment l'option des tables MyISAM `DELAYED KEYS` doit être utilisè. Voir Section 5.5.2 [Server parameters], page 390.

`--delay-key-write-for-all-tables; En MySQL 4.0.3 vous devez utiliser --delay-key-write=ALL á la place.`
: Ne vide pas les buffers de clès entre deux ècritures pour les tables `MyISAM`. Voir Section 5.5.2 [Server parameters], page 390.

`--des-key-file=filename`
: Lit les clès par dèfaut utilisèes par `DES_ENCRYPT()` et `DES_DECRYPT()` dans ce fichier.

`--enable-external-locking (was --enable-locking)`
: Active le verrouillage systéme. Notez que si vous utilisez cette option sur un systéme pour qui `lockd` ne fonctionne pas (comme Linux), vous allez bloquer rapidement mysqld avec les verrous.

`--enable-named-pipe`
: Active le support des tunnels nommès (seulement sur NT/Win2000/XP).

`-T, --exit-info`
: Cette option est la combinaison d'options que vous pouvez utiliser pour le dèbogage du serveur mysqld; Si vous ne savez pas ce que ca fait exactement, ne les utilisez pas !

`--flush` Ecrit toutes les donnèes sur le disque aprés chaque requête SQL. Normalement, MySQL fait des ècritures sur le disque aprés chaque requête, et laisse le systéme d'exploitation assurer la synchronisation avec le disque. Voir Section A.4.1 [Crashing], page 697.

`-?, --help`
: Affiche l'aide courte et termine le programme.

`--init-file=file`
: Lit les commandes SQL dans ce fichier au dèmarrage.

`-L, --language=...`
: Spècifie la langue utilisèe pour les messages d'erreur du client. Le chemin complet doit être utilisè. Voir Section 4.6.2 [Languages], page 287.

`-l, --log[=file]`
: Enregistre les connexions et les requêtes dans ce fichier. Voir Section 4.9.2 [Query log], page 328.

`--log-isam[=file]`
: Enregistre toutes les modifications des tables ISAM/MyISAM dans ce fichier (uniquement nècessaire pour dèboguer ISAM/MyISAM).

`--log-slow-queries[=file]`
: Enregistre toutes les requêtes qui ont pris plus de `long_query_time` secondes a s'exècute, dans ce fichier. Voir Section 4.9.5 [Slow query log], page 331.

`--log-update[=file]`
: Enregistre les modifications dans le fichier `file.#` oú `#` est un nombre unique si il n'est pas prècisè. Voir Section 4.9.3 [Update log], page 328.

`--log-long-format`
: Enregistre des informations supplèmentaires dans le fichier d'historique. Si vous utilisez l'option `--log-slow-queries`, alors les requêtes que vous qui n'utilisent pas les index sont enregistrèes dans le log de requêtes longues.

`--low-priority-updates`
: Les opèrations de modifications de table (`INSERT`/`DELETE`/`UPDATE`) auront une prioritè infèrieure aux sèlections. Cela peut être aussi fait via l'attribut `{INSERT | REPLACE | UPDATE | DELETE} LOW_PRIORITY ...` pour baisser la prioritè d'une requête, ou avec `SET LOW_PRIORITY_UPDATES=1` pour changer la prioritè dans plus d'un thread. Voir Section 5.3.2 [Table locking], page 381.

`--memlock`
: Verrouille le processus `mysqld` en mèmoire. Cela fonctionne si votre systéme support la fonction `mlockall()` (comme Solaris). Ceci peut être utile si vous avez des problémes avec le systéme d'exploitation qui force `mysqld` a utiliser le swap sur le disque.

`--myisam-recover [=option[,option...]]]`
: Cette option est la combinaison de `DEFAULT`, `BACKUP`, `FORCE` et `QUICK`. Vous pouvez aussi lui donner la valeur explicite de `""` si vous voulez dèsactiver cette option. Si cette option est utilisèe, `mysqld` va vèrifier si la table est marquèe comme corrompue á l'ouverture de chaque table (cette derniére option ne fonctionne que si vous utilisez l'option `--skip-external-locking`). Si c'est le cas, `mysqld` va essayer de vèrifier la table. Si la table ètait corrompue, `mysqld` essaie alors de la rèparer.

  L'option suivante va affecter la maniére avec la quelle la rèparation s'effectue.

Option	Description
DEFAULT	Identique á ne pas donner d'option á `--myisam-recover`.
BACKUP	Si la table a ètè modifièe durant la rèparation, sauver une copie du fichier '`table_name.MYD`', sous le nom de '`table_name-datetime.BAK`'.
FORCE	Exècute une rèparation même si nous allons perdre une ou plusieurs lignes dans le fichier .MYD.

QUICK Ne vèrifie pas les lignes dans la table si il n'y a pas eu d'effacement.

Avant que la table ne soit automatiquement rèparèe, MySQL va ajouter une note dans le fichier de log d'erreurs. Si vous voulez être capable de restaurer la plupart des erreurs sans intervention de l'utilisateur, il vaut utiliser les options `BACKUP,FORCE`. Cela va forcer la rèparation de la table, même si quelques lignes sont effacèes, et conserve le vieux fichier de donnèes comme sauvegarde, pour examen ultèrieur.

`--pid-file=path`
Le chemin jusqu'au fichier de pid utilisè par `safe_mysqld`.

`-P, --port=...`
Numèro de port utilisè pour attendre les connexion TCP/IP.

`-o, --old-protocol`
Utilise le protocole 3.20, pour la compatibilitè avec de trés vieux clients. Voir Section 2.5.4 [Upgrading-from-3.20], page 111.

`--one-thread`
Utilise uniquement un thread (pour dèbogage sous Linux). Voir Section E.1 [Debugging server], page 815.

`-O, --set-variable var=option`
Donne une valeur á une variable. `--help` liste ces variables. Vous pouvez trouver une description compléte des variables dans la section sur la commande `SHOW VARIABLES` de ce manuel. Voir Section 4.5.6.4 [SHOW VARIABLES], page 273. La section de paramètrage inclut des informations sur comment exploiter ces variables. Notez que `--set-variable` est obsoléte depuis MySQL 4.0, utilisez simplement la syntaxe `--var=option`. Voir Section 5.5.2 [Server parameters], page 390.

En MySQL 4.0.2, il est possible de changer une variable directement avec la syntaxe `--variable-name=option` et `set-variable` n'est plus nècessaire dans le fichier de configuration.

Si vous voulez restreindre la valeur maximale que peut prendre une option via la commande `SET`, vous pouvez dèfinir une limite en utilisant l'option de ligne de commande `--maximum-variable-name`. Voir Section 5.5.6 [SET OPTION], page 396.

Notez que lorsque vous donnez une valeur á une variable, MySQL peut corriger automatiquement votre valeur pour rester dans un intervalle donnè, et ajuster un peu la valeur pour qu'elle soit optimale.

`--safe-mode`
Ignore certains ètapes d'optimisation.

`--safe-show-database`
Avec cette option, la commande `SHOW DATABASES` retourne uniquemnt les bases pour lesquelles l'utilisateur a des droits. Depuis la version 4.0.2, cette option est abandonnèe, et ne fait plus rien (l'option est activèe par dèfaut) car nous avons dèsormais le droit de `SHOW DATABASES`. Voir Section 4.3.1 [GRANT], page 226.

`--safe-user-create`
: Si cette option est activèe, un utilisateur ne peut pas crèer de nouveaux utilisateurs avec la commande GRANT si l'utilisateur n'a pas les droits de `INSERT` dans la table `mysql.user` ou dans aucune colonne de cette table.

`--skip-bdb`
: Dèsactive l'utilisation des tables BDB. Cela va èconomiser de la mèmoire et accèlèrer le serveur un peu.

`--skip-concurrent-insert`
: Dèsactive la possibilitè de sèlectionner et insèrer en même temps dans les tables `MyISAM` (cela n'est utile que si vous pensez que vous avez trouvè un bug dans cette fonctionnalitè).

`--skip-delay-key-write; En MySQL 4.0.3, il est recommandè d'utiliser l'option`
: –delay-key-write=OFF á la place. Ignorez l'option `DELAY_KEY_WRITE` de toutes les tables. Voir Section 5.5.2 [Server parameters], page 390.

`--skip-grant-tables`
: Cette option force le serveur á ne pas utiliser le systéme de privilége du tout. Cela donne á tous l'**accés complet** á toutes les bases de données ! Vous pouvez demander á un serveur en exècution d'utiliser á nouveau les tables de droits en exècutant la commande `mysqladmin flush-privileges` ou `mysqladmin reload`).

`--skip-host-cache`
: Ne pas utiliser le cache de nom de domaine pour une rèsolution des IP plus rapide, mais interroger le serveur DNS á chaque connexion. Voir Section 5.5.5 [DNS], page 395.

`--skip-innodb`
: Dèsactive l'utilisation des tables Innodb. Cela va èconomiser de la mèmoire et accèlèrer le serveur un peu.

`--skip-external-locking (ancien --skip-locking)`
: Ne pas utiliser le verrouillage du systéme. Pour utiliser les utilitaires `isamchk` ou `myisamchk` vous devez alors èteindre le systéme. Voir Section 1.2.3 [Stability], page 8. Notez qu'en MySQL version 3.23 vous pouvez utiliser la commande `REPAIR` et `CHECK` pour rèparer ou vèrifier des tables `MyISAM` tables.

`--skip-name-resolve`
: Les noms d'hôtes ne sont pas rèsolus. Toutes les colonnes `Host` dans vos tables de droits doivent être des IP numèriques ou le mot `localhost`. Voir Section 5.5.5 [DNS], page 395.

`--skip-networking`
: Ne pas attendre les connexions TCP/IP du tout. Toutes les interactions du serveur `mysqld` seront faites avec les sockets Unix. Cette option est particuliérement recommandèe pour les systémes qui utilisent des requêtes locales. Voir Section 5.5.5 [DNS], page 395.

`--skip-new`
: Ne pas utiliser les nouvelles routines qui sont possiblement erronèes.

`--skip-symlink`
: Ne pas effacer ou renommer les fichiers qui ont un lien symbolique dans le dossier de donnèes.

`--skip-safemalloc`
: Si MySQL est configurè avec `--with-debug=full`, tous les programmes vèrifieront la mèmoire pour rechercher les ècrasment de zone lors des allocations et libèrations de mèmoire. Comme ce test est lent, vous pouvez l'èviter, si vous n'avez pas besoin de tester la mèmoire, en utilisant cette option.

`--skip-show-database`
: Ne pas autoriser la commande `SHOW DATABASES`, a moins que l'utilisateur n'ait les droits de `SHOW DATABASES`. Depuis la version 4.0.2, vous n'avez plus besoin de cette option, car les droits pour ce faire sont distribuès avec le droit de `SHOW DATABASES`.

`--skip-stack-trace`
: Ne pas ècrire les piles de traces. Cette option est pratique lorsque vous utilisez `mysqld` avec un dèbogueur. Sur certains systémes, vous devez aussi utiliser cette option pour obtenir un fichier de core. Voir Section E.1 [Debugging server], page 815.

`--skip-thread-priority`
: Dèsactive la priorisation des threads pour amèliorer la vitesse de rèponse.

`--socket=path`
: Le fichier de socket á utiliser pour les connexions locales, au lieu du fichier par dèfaut `/tmp/mysql.sock`.

`--sql-mode=option[,option[,option...]]`
: Cette option peut être la combinaison de : `REAL_AS_FLOAT`, `PIPES_AS_CONCAT`, `ANSI_QUOTES`, `IGNORE_SPACE`, `SERIALIZE` et `ONLY_FULL_GROUP_BY`. Ellep peut aussi être vide (`""`) si vous voulez remettre cette option á 0.

  En spècifiant toutes les options ci-dessus, vous obtiendrez le même effet qu'avec l'option –ansi. Avec cette option, vous pouvez activer les modes SQL dont vous avez besoin. Voir Section 1.7.2 [ANSI mode], page 33.

`--temp-pool`
: En utilisant cette option, vous allez rèduire le jeu de noms qui sont utilisès lors de la crèation de fichier temporaires, plutôt qu'un nom unique á chaque fois. Ceci est un palliatif au noyau Linux qui crée plusieurs fichiers nouveaux avec des noms diffèrents. Avec l'ancien comportement, Linux semble 'perdre de la mèmoire', car ils sont allouès au cache d'entrèes du dossier au lieu de celui du disque.

`--transaction-isolation= { READ-UNCOMMITTED | READ-COMMITTED | REPEATABLE-READ | SERIALIZABLE }`
: Configure le niveau d'isolation des transactions. Voir Section 6.7.3 [SET TRANSACTION], page 521.

`-t, --tmpdir=path`
: Chemin vers les fichiers temporaires. Il peut s'avèrer pratique si votre dossier par dèfaut `/tmp` rèside dans une partition qui est trop petite pour absorber les tables temporaires.

`-u, --user= [user_name | userid]`
: Exècute le dèmon `mysqld` avec l'utilisateur `user_name` ou `userid` (numèrique). Cette option est **obligatoire** lorsque vous dèmarrez `mysqld` en tant que root.

`-V, --version`
: Affiche les informations de version.

`-W, --log-warnings (Was --warnings)`
: Enregistre les alertes comme `Aborted connection...` dans le fichier '`.err`'. Voir Section A.2.9 [Communication errors], page 690.

Il est possible de modifier la plupart des valeurs durant l'exècution, avec la commande `SET` command. Voir Section 5.5.6 [SET OPTION], page 396.

4.1.2 Fichier d'options '`my.cnf`'

MySQL peu, depuis la version 3.22, lire des options de dèmarrage par dèfaut pour le serveur en ligne de commande, et, par le client, dans un fichier.

MySQL lit les options par dèfaut dans les fichiers suivants sous Unix :

Fichier	Objet
`/etc/my.cnf`	Options globales
`DATADIR/my.cnf`	Options spècifiques au serveur
`defaults-extra-file`	Le fichier spècifiè par –defaults-extra-file=#
`~/.my.cnf`	Options spècifiques á l'utilisateur

`DATADIR` est le dossier de donnèes de MySQL (typiquement '`/usr/local/mysql/data`' pour les installation binaires ou '`/usr/local/var`' pour une installation source). Notez que c'est ce dossier qui a ètè spècifiè au moment de la configuration et non pas le dossier de l'option `--datadir` lorsque `mysqld` dèmarre ! (`--datadir` n'a aucun effet sur le serveur, car le serveur recherche les donnèes avant de traiter les options de ligne de commande).

MySQL lit les fichiers d'options suivants sous Windows :

Fichier	Contenu
`windows-system-directory\my.ini`	Options globales
`C:\my.cnf`	Options globales

Notez que sous Windows, vous devez spècifier les chemins avec `/` plutôt que `\`. Si vous utilisez `\`, vous devez le spècifier deux fois, car `\` est un caractére de protection pour MySQL.

MySQL essaie de lire les fichiers d'options dans l'ordre dans lequel ils sont prèsentès ci-dessus. Si des options sont spècifièes plusieurs fois, la derniére occurrence utilisèe prend la prèsèeance sur les options spècifièes avant. Les options de ligne de commande ont la prioritè sur les options spècifièes dans les fichiers. Certaines options peuvent être spècifièes en utilisant des variables d'environnement. Les options spècifièes en ligne de commande ou

en fichier ont la prioritè sur les options qui le sont via une variable d'environnement. Voir Annexe F [Environment variables], page 828.

Les programmes suivants utilisent les fichiers d'options : `mysql`, `mysqladmin`, `mysqld`, `mysqld_safe`, `mysql.server`, `mysqldump`, `mysqlimport`, `mysqlshow`, `mysqlcheck`, `myisamchk` et `myisampack`.

Toute option longue qui doit être spècifiée en ligne de commande lorsque MySQL fonctionne, peut être aussi configurèe dans le fichier d'options (sans les doubles tirets). Exècutez le programme avec l'option `--help` pour avoir une liste des options disponibles.

Un fichier d'options contient des lignes ayant la forme suivante :

`#comment` Les lignes de commentaires commencent avec '`#`' ou '`;`'. Les lignes vides sont ignorèes.

`[group]` `group` est le nom du programme ou du groupe pour lequel vous souhaitez configurer des options. Aprés une ligne de groupe, toutes les `option` et `set-variable` s'appliqueront au groupe nommè, jusqu'á la fin du fichier d'option ou du dèmarrage d'un autre groupe.

`option` Ceci est èquivalent á `--option` sur la ligne de commande.

`option=value`
Ceci est èquivalent á `--option=value` sur la ligne de commande.

`set-variable = variable=value`
Ceci est èquivalent á `--set-variable variable=value` sur la ligne de commande. Cette syntaxe doit être utilisèe pour spècifier la valeur d'une variable `mysqld`. Notez que `--set-variable` est obsoléte depuis MySQL 4.0, utilisez simplement `--variable=value` comme tel.

Le groupe `client` vous permet de spècifier des options qui ne s'appliquent qu'aux clients MySQL et non pas au serveur `mysqld`. C'est le groupe idèal pour spècifier des mots de passe de connexion au serveur (mais assurez-vous que vous êtes le seul á accèder á ce fichier !!).

Notez que pour les options et les valeurs, tous les caractéres blancs de dèbut et de fins seront automatiquement effacès. Vous pouvez utiliser les sèquences d'èchappement '`\b`', '`\t`', '`\n`', '`\r`', '`\\`' et '`\s`' dans votre chaîne á la place ('`\s`' == espace).

Voici un exemple typique de fichier d'options globales :

```
[client]
port=3306
socket=/tmp/mysql.sock

[mysqld]
port=3306
socket=/tmp/mysql.sock
set-variable = key_buffer_size=16M
set-variable = max_allowed_packet=1M

[mysqldump]
quick
```

Voici un exemple typique de fichier d'options utilisateur :

```
[client]
# Le mot de passe suivant va être utilisè avec le serveur
password=mon_mot_de_passe

[mysql]
no-auto-rehash
set-variable = connect_timeout=2

[mysqlhotcopy]
interactive-timeout
```

Si vous avez une distribution source, vous trouverez des exemples de configuration dans les fichiers nommès '`my-xxxx.cnf`' dans le dossier '`support-files`'. Si vous avez une distribution binaire, regardez dans le dossier '`DIR/support-files`', oú `DIR` est le chemin de l'installation MySQL (typiquement '`/usr/local/mysql`'). Actuellement, il y a des exemples de configuration pour des systémes petits, moyens, grands et trés grands. Vous pouvez copier l'un des fichiers '`my-xxxx.cnf`' dans votre dossier utilisateur (renommez le fichier en '`.my.cnf`') pour le tester.

Tous les clients MySQL qui supportent les fichiers d'options, acceptent les options suivantes :

Option	**Description**
–no-defaults	Ne lire aucun fichier d'options.
–print-defaults	Affiche le nom du programme et toutes les options qui s'y trouvent.
–defaults-file=full-path-to-default-file	Utilise uniquement le fichier de configuraiton donnè.
–defaults-extra-file=full-path-to-default-file	Lit ce fichier de configuration aprés le fichier de configuration global, mais avant le fichier de configuration utilisateur.

Notez ques les options ci-dessus doivent être en ligne de commande pour être utilisèes ! `--print-defaults` peut quand même être utilisè directement aprés la commande `--defaults-xxx-file`.

Note pour les dèveloppeurs : la gestion des fichiers d'options est implèmentèe simplement en traitant toutes les options qui correspondent (c'est-á-dire, toutes les options approprièes du groupe), avant les arguments de ligne de commande. Cela fonctionne bien pour les programmes qui utilisent la derniére occurrence comme valeur d'option, si elle est spècifièe plusieurs fois. Si vous avez un vieux programme qui traite les options multiples de cette façon mais ne lit pas les fichiers d'options, vous n'avez besoin que de deux lignes pour qu'il accepte cette fonctionnalitè. Rècupèrez le code source de n'importe quel client MySQL standard pour voir comment le faire.

En scripts shell, vous pouvez utiliser la commande '`my_print_defaults`' pour analyser les fichiers de configuration :

```
shell> my_print_defaults client mysql
--port=3306
```

```
--socket=/tmp/mysql.sock
--no-auto-rehash
```

La ligne ci-dessus affiche toutes les options pour les groupes 'client' et 'mysql'.

4.1.3 Installer plusieurs serveurs sur la même machine

Dans certains cas, vous aurez besoin de plusieurs dèmons `mysqld` sur la même machine. Vous pouvez, par exemple, faire tourner une vieille version de MySQL pour la tester avec une nouvelle. Vous pouvez aussi donner des accés diffèrents á des utilisateurs de diffèrents serveurs `mysqld`, qu'il gérent eux-mêmes.

Une mèthode pour avoir plusieurs serveurs diffèrents sur la même machine est de le configurer avec diffèrents sockets et ports comme suit :

```
shell> MYSQL_UNIX_PORT=/tmp/mysqld-new.sock
shell> MYSQL_TCP_PORT=3307
shell> export MYSQL_UNIX_PORT MYSQL_TCP_PORT
shell> scripts/mysql_install_db
shell> bin/safe_mysqld &
```

L'annexe sur les variables d'environnement inclut une liste des variables d'environnement que vous pouvez utiliser pour paramètrer `mysqld`. Voir Annexe F [Environment variables], page 828.

La mèthode ci-dessus est immèdiate et peu propre pour ceux qui font des tests. Ce qui est bien avec cette mèthode, c'est que les connexions que vous rèalisez avec le shell ci-dessus seront automatiquement redirigèes vers le serveur en fonctionnement.

Si vous avez besoin d'une mèthode plus permanente, il est recommandè de crèer un fichier d'options pour chaque serveur. Voir Section 4.1.2 [Option files], page 198. Dans votre script de dèmarrage du serveur, vous pourriez spècifier tous les serveurs :

```
safe_mysqld --defaults-file=path-to-option-file
```

Enfin, les options suivantes doivent être diffèrentes pour chaque serveur :

- port=#
- socket=path
- pid-file=path

Les options suivantes doivent être diffèrentes, si elles sont utilisèes :

- log=path
- log-bin=path
- log-update=path
- log-isam=path
- bdb-logdir=path

Si vous voulez plus de performances, vous pouvez aussi configurer diffèremment les options suivantes :

- tmpdir=path
- bdb-tmpdir=path

Voir Section 4.1.1 [Command-line options], page 192.

Si vous installez une version binaire de MySQL (fichiers .tar) et que vous les dèmarrez avec `./bin/safe_mysqld`, alors dans la plupart des cas, la seule option que vous devez modifier est la socket `socket` et le port `port` dans le script `safe_mysqld`.

Voir Section 4.1.4 [Running Multiple MySQL Servers on the Same Machine], page 202.

4.1.4 Faire fonctionner plusieurs serveurs MySQL sur la même machine

Il y a des situations oú vous souhaiterez avoir plusieurs serveurs MySQL sur la même machine. Par exemple, si vous voulez tester une nouvelle version du serveur avec votre configuration de production sans perturber votre installation de production. Ou bien, vous êtes un fournisseur de services Internet, et vous voulez fournir des installations distinctes pour des clients diffèrents.

Si vous voulez exècuter plusieurs serveurs MySQL, le plus simple et de compiler les serveurs avec diffèrents ports TCP/IP et fichiers de sockets pour qu'ils ne soient pas tous á l'ècoute du même port ou de la même socket. Voir Section 4.7.3 [`mysqld_multi`], page 293.

Supposons que votre serveur est configurè avec le numèro de port par dèfaut, et le numèro de socket par dèfaut. Alors vous pouvez configurer le nouveau serveur avec la commande `configure` suivante :

```
shell> ./configure  --with-tcp-port=numero_port \
             --with-unix-socket-path=nom_fichier \
             --prefix=/usr/local/mysql-3.22.9
```

Ici, les options `numero_port` et `nom_fichier` doivent être diffèrentes des valeurs par dèfaut, et avec l'option `--prefix` vous allez spècifier un autre dossier d'installation que celui qui est dèjá utilisè par la premiére installation.

Vous pouvez vèrifier le nom de la socket qui est utilisèe par un serveur MySQL avec cette commande :

```
shell> mysqladmin -h hostname --port=numero_port variables
```

Notez que si vous spècifiez "`localhost`" comme nom d'hôte, `mysqladmin` va utiliser les sockets Unix plutôt que TCP/IP.

Si vous avez un serveur MySQL qui utilise dèjá le port, vous obtiendrez une liste des variables de configuration les plus importantes de MySQL, y compris le nom de la socket.

Vous n'avez pas á recompiler un nouveau serveur MySQL juste pour le dèmarrer sous un autre port et une autre socket. Vous pouvez changer le port et la socket utilisèe au moment du dèmarrage, avec `safe_mysqld` :

```
shell> /path/to/safe_mysqld --socket=nom_fichier --port=numero_port
```

`mysqld_multi` peut aussi prendre `safe_mysqld` (ou `mysqld`) comme argument, et passer les options depuis un fichier de configuration á `safe_mysqld`, et ainsi, á `mysqld`.

Si vous exècutez un nouveau serveur sur les mêmes donnèes qu'un autre serveur, avec le log activè, vous devez spècifier le nom du fichier de log á `safe_mysqld` avec l'option `--log`, `--log-update`, ou `--log-slow-queries`. Sinon, les deux serveurs risquent d'ècrire dans le même fichier de log.

Attention : normalement, vous ne devez pas avoir deux serveurs qui modifient en même temps les données dans les mêmes bases. Si votre OS ne supporte pas le verrouillage sans èchec, cela peut vous mener á de dèplaisantes surprises !

Si vous voulez utiliser un autre dossier de fichiers pour le second serveur, vous pouvez utiliser l'option `--datadir=path` de `safe_mysqld`.

Notez aussi que dèmarrer plusieurs serveurs MySQL (`mysqlds`) dans diffèrentes machines, et les laisser accèder aux même données via `NFS` est gènèralement une **mauvaise idèe** ! Le probléme est que `NFS` va devenir un frein au niveau de la vitesse. Il n'est pas destinè á une telle utilisation. Et de plus, vous devrez trouver une solution pour vous assurer que deux des `mysqld` n'interférent pas entre eux. Actuellement, il n'y a pas de plate-forme qui soit 100% fiable pour le verrouillage de fichiers (le dèmon `lockd`), dans toutes les situations. Pourtant, il va y avoir un risque supplèmentaire avec `NFS` ; cela rendra le travail encore plus compliquè pour le dèmon `lockd`. Alors, simplifiez-vous la vie, et oubliez cela. La solution efficace est d'avoir un ordinateur avec un systéme d'exploitation, qui gére efficacement les threads et a plusieurs processeurs.

Lorque vous voulez vous connecter au serveur MySQL qui fonctionne avec un autre port que le port qui est compilè par dèfaut dans votre client, vous pouvez utiliser l'une des mèthodes suivantes :

- Lancez le client avec l'option `--host 'hostname' --port=numero_port` pour vous connecter en TCP/IP, ou `[--host localhost] --socket=nom_fichier` pour vous connecter via une socket Unix.
- Dans vos programmes C ou Perl, vous pouvez indiquer le port ou la socket lors de la connexion au serveur MySQL.
- Si vous utilisez le code Perl avec le module `DBD::mysql`, vous pouvez lire les options des fichiers d'options MySQL. Voir Section 4.1.2 [Option files], page 198.

```
$dsn = "DBI:mysql:test;mysql_read_default_group=client;
        mysql_read_default_file=/usr/local/mysql/data/my.cnf"
$dbh = DBI->connect($dsn, $user, $password);
```

- Modifiez les variables d'environnement `MYSQL_UNIX_PORT` et `MYSQL_TCP_PORT` pour qu'elles pointent sur la socket Unix et le port TCP/IP voulu avant de dèmarrer vos clients. Si vous utilisez une socket ou un port spècifique, il est recommandè de modifier ces variables dans votre fichier '`.login`'. Voir Annexe F [Environment variables], page 828.
- Specifiez la socket et le port TCP/IP par dèfaut dans le fichier '`my.cnf`' de votre dossier personnel. Voir Section 4.1.2 [Option files], page 198.

4.2 Régles de sècuritè et droits d'accés au serveur MySQL

MySQL est pourvu d'un systéme avancè mais non standard de droits. Cette section dècrit son fonctionnement.

4.2.1 Instructions gènèrales de sècuritè

Tous ceux qui utilisent MySQL avec un serveur connectè á l'internet doivent lire cette section, pour èviter les erreurs de sècuritè les plus courantes.

En parlant de sècuritè, nous mettons l'accent sur la nècessitè de protèger la totalitè du serveur (et non pas seulement le serveur MySQL), contre tous les types d'attaques possibles : cheval de troye, virus, dènis de services, ècoute èlectronique. Nous ne couvrirons pas la totalitè des aspects et des pannes ici.

MySQL utilise un systéme de sècuritè basè sur les listes de contrôle d'accés (Access Control Lists; ACLs) pour toutes les connexions, requêtes et autres opèrations qu'un utilisateur peut tenter d'exècuter. Il y a aussi le support des connexions chiffrèes par mode SSL, entre le client et le serveur. De nombreux concepts sont prèsentès ici, et ne sont pas spècifiques á MySQL. Les mêmes concepts s'appliquent á de nombreuses applications.

Lorsque vous faîtes fonctionner MySQL, suivez ces régles le plus souvent possible :

- **Ne jamais donner á quiconque (hormis l'utilisateur mysql root) l'accés á la table `user` dans la base `mysql`!** Cette table est primordiale. **Le mot de passe chiffrè stockè dans la table est le vèritable mot de passe MySQL**. Quiconque connaît le mot de passe stockè dans la table `user` et a accés á la liste des hôtes d'un compte **peut facilement se connecter avec cet utilisateur**.
- Connaissez á fond le systéme de droits de MySQL. Les commandes `GRANT` et `REVOKE` sont utilisèes pour contrôler les accés á MySQL. Ne donnez pas plus de droits que nècessaire. Ne donnez jamais des droits á tous les hôtes.

  Liste de points á vèrifier :

  - Essayez `mysql -u root`. Si vous êtes capable de vous connecter avec succés au serveur sans qu'un mot de passe n'ai ètè demandè, vous avez un probléme. Toute personne peut se connecter au serveur MySQL en tant qu'utilisateur `root` avec tous les droits ! Passez en revue votre installation MySQL, en faisant attention aux mots de passe `root`.
  - Utilisez la commande `SHOW GRANTS` et vèrifiez qui a accés á quoi. Supprimez les droits qui ne sont pas nècessaires avec la commande `REVOKE`.
- Ne gardez par de mot de passe en clair dans vos tables. Lorsque votre ordinateur est infiltrè, l'intrus pourra alors obtenir la liste compléte des mots de passe, et les utiliser. Au lieu de cela, utilisez les fonctions `MD5()`, `SHA1()` ou une autre fonction de hash.
- Ne choisissez pas vos mots de passe dans un dictionnaire. Il y a des programmes spèciaux qui sont capable de les casser. Même des mots de passe comme "xfish98" sont trés faibles. Bien meilleur est "duag98" qui contient aussi le mot "fish" mais tapè avec une touche á gauche sur le clavier standard QWERTY. Une autre mèthode est d'utiliser "Pfspeb" qui est un mot crèè á partir des premiéres lettres de la phrase "Papa fume sa pipe en bois". C'est un mot de passe facile á retenir, difficile á deviner pour une personne qui ne le connaît pas.
- Investissez dans un pare-feu. Cela vous protége de presque 50 % de tous les types de trous de sècuritè des logiciels. Placez MySQL derriére un pare-feu, ou dans une zone dèmilitarisèe (DMZ).

  Liste de points á vèrifier :

  - Essayez de scanner vos ports depuis Internet, en utilisant des outils comme `nmap`. MySQL utilise le port 3306 par dèfaut. Ce port doit être inaccessible depuis des hôtes indus. Un autre moyen simple de tester si votre port MySQL est accessible ou pas, est d'essayer la commande suivante depuis une machine distante, oú `server_host` est l'hôte de votre serveur MySQL :

```
shell> telnet server_host 3306
```

Si vous obtenez une connexion et des caractéres ètranges, c'est que le port est ouvert, et qu'il devrait être fermè par votre routeur ou votre pare-feu, á moins que vous n'ayez une bonne raison de le garder ouvert. Si `telnet` attend ou que la connexion est refusèe, c'est que tout est bon : le port est bloquè.

- Ne faites jamais aucune confiance aux donnèes des utilisateurs. Ils peuvent essayer de nombreux trucs pour vous envoyer du code spècial ou faire passer des sèquences de caractéres protègès via des formulaires web, ou n'importe quel outil que votre application utilise. Soyez certain que votre application reste sècurisèe même si un utilisateur obtient du code comme "`; DROP DATABASE mysql;`". Ceci est un exemple extrême, mais de grosse fuites de donnèes ou des pertes peuvent survenir si un pirate utilise des techniques similaires, et que vous n'êtes pas prèparè pour.

  Vèrifiez aussi toutes les donnèes numèriques. Une erreur commune est de protèger uniquement les chaînes. Certains pensent que si une base contient des donnèes publiques, elle ne doit pas être protègèe. C'est totalement faux. Au minimum, un attaque en dènis de service peut être pratiquèe sur ces bases. Le plus simple moyen pour vous protèger contre ce type d'attaque est d'utiliser des apostrophes autour des constantes numèriques : `SELECT * FROM table WHERE ID='234'` plutôt que `SELECT * FROM table WHERE ID=234`. MySQL convertit automatiquement la chaîne en nombre et supprime tous les caractéres non numèriques avant insertion.

  Point á vèrifier :

  – Toutes les applications web :
    - Essayez d'entrer des guillemets '' et '"' dans tous vos formulaires web. Si vous obtenez une erreur MySQL, alors ètudiez le probléme immèdiatement.
    - Essayez de modifier toute URL dynamique en ajoutant les codes `%22` ('"'), `%23` ('#') et `%27` ('') dedans.
    - Essayez de modifier les types de donnèes des URL dynamiques, de numèrique en chaîne, grâce aux exemples ci-dessus. Votre application doit être sècurisèe contre ce type d'attaques.
    - Essayez d'entrer des caractéres comme des espaces ou des symboles spèciaux au lieu des nombres, dans les champs numèriques. Votre application doit les supprimer avant de le passer á MySQL, ou votre application doit gènèrer une erreur. Passer des valeurs non vèrifièes á MySQL est trés dangereux.
    - Vèrifiez la taille des donnèes avant de les passer á MySQL.
    - Donnez un autre nom d'utilisateur et un autre mot de passe á votre application que votre nom d'utilisateur d'administration. Ne donnez pas á votre application plus de droits que nècessaire.
  – Utilisateurs de PHP :
    - Etudiez la fonction `addslashes()`. Depuis PHP 4.0.3, la fonction `mysql_escape_string()` est disponible avec la même fonctionnalitè que son alter ego en API MySQL C.
  – Utilisateurs de l'API MySQL C :
    - Vèrifiez les appels á `mysql_real_escape_string()`.
  – Utilisateurs de MySQL++ :

    - Vèrifiez les options `escape` et `quote` des requêtes.
  - Utilisateurs Perl DBI :
    - Vèrifiez la mèthode `quote()` ou utilisez des variables liées.
  - Utilisateurs de Java JDBC :
    - Utilisez l'objet `PreparedStatement` et les variables liées.
- Ne transmettez aucunes données non chiffrèes via Internet. Ces données peuvent être accessibles á quiconque a le temps et la capacitè d'intercepter des données et de les utiliser pour son propre usage. Au lieu de cela, utilisez un protocole chiffrè tel que SSL ou SSH. MySQL supporte les connexions SSL en interne, depuis la version 4.0.0. Le forward de port de SSH peut être utilisè pour crèer un tunnel chiffrè et compressè pour les communications.
- Apprenez á utiliser les utilitaires `tcpdump` et `strings`. Pour la plupart des cas, vous pouvez vèrifier si les flux de donnèes MySQL sont chiffrès en utilisant une commande telle que celle ci :

  ```
  shell> tcpdump -l -i eth0 -w - src or dst port 3306 | strings
  ```

  (Cela fonctionne sous Linux et devrait aussi fonctionner sous les autres systémes avec de lègéres adaptations). Attention : si vous ne voyez pas de donnèes, cela ne signifie pas que vous avez des donnèes chiffrèes. Si vous avez besoin de forte sècuritè, vous devriez consulter un expert de la sècuritè.

4.2.2 Comment protèger MySQL contre les pirates

Lorsque vous vous connectez á MySQL, vous devriez avoir besoin d'un mot de passe. Ce mot de passe n'est pas transmis en texte clair sur le rèseau, mais comme l'algorithme de chiffrement n'est pas trés fort, quelques efforts permettront á un pirate d'obtenir votre mot de passe s'il est capable de surveiller le trafic entre le client et le serveur. Si la connexion entre le client et le serveur utilise des rèseaux non fiables, il est alors recommandè d'utiliser un tunnel SSH.

Toutes les autres informations sont transfèrèes comme du texte clair, et quiconque surveille la connexion pourra les lire. Si vous souhaitez relever ce niveau de sècuritè, il est recommandè d'utiliser le protocole compressè (avec les versions de MySQL 3.22 et plus rècentes), pour compliquer considèrablement le probléme. Pour rendre la communication encore plus sûre, vous pouvez aussi utiliser `ssh`. Vous trouverez une version `Open Source` du client `ssh` sur le site `http://www.openssh.org/`, et une version commerciale du client `ssh` sur le site de `http://www.ssh.com/`. Avec eux, vous pouvez mettre en place une connexion TCP/IP chiffrèe entre le serveur et le client MySQL.

Si vous utilisez MySQL 4.0, vous pouvez aussi utiliser le support OpenSSL interne. Voir Section 4.3.9 [Secure connections], page 239.

Pour rendre le systéme MySQL encore plus sûr, nous vous recommandons de suivre les suggestions suivantes :

- Utilisez des mots de passe pour tous les utilisateurs MySQL. N'oubliez pas que tout le monde peut se connecter avec un nom d'utilisateur quelconque, simplement avec l'option `mysql -u autre_utilisateur nom_de_base`, si `autre_utilisateur` n'a pas de mot de passe. C'est un comportement classique pour les applications client/serveur

que le client spècifie son nom d'utilisateur. Vous pouvez modifier les mots de passe de tous les utilisateurs en modifiant le script `mysql_install_db` avant de l'exècuter, ou vous pouvez modifier seulement le mot de passe du `root` MySQL comme ceci :

```
shell> mysql -u root mysql
mysql> UPDATE user SET Password=PASSWORD('nouveau_mot_de_passe')
    ->             WHERE user='root';
mysql> FLUSH PRIVILEGES;
```

- N'exècutez jamais le dèmon MySQL avec l'utilisateur Unix `root`. C'est trés dangeureux, car tout personne ayant le droit de `FILE` pour crèer des fichiers au nom du `root` (par exemple, `~root/.bashrc`). Pour èviter cela, `mysqld` refusera de s'exècuter au nom de `root` á moins que soit prècisè l'option `--user=root`.

  `mysqld` peut être exècutè avec un utilisateur ordinaire sans droits particuliers. Vous pouvez aussi crèer un utilisateur Unix `mysql` pour rendre cela encore plus sûr. Si vous exècutez `mysqld` sous un autre utilisateur Unix, vous n'avez pas á changer le mot de passe `root` dans la table `user`, car les noms d'utilisateurs MySQL n'ont rien á voir avec les noms d'utilisateurs Unix. Pour dèmarrer `mysqld` sous un autre nom d'utilisateur Unix, ajoutez la ligne `user`, qui spècifie le nom de l'utilisateur, dans le fichier d'options de `[mysqld]` '`/etc/my.cnf`' ou dans le fichier '`my.cnf`' prèsent dans le dossier de donnèes du serveur. Par exemple :

  ```
  [mysqld]
  user=mysql
  ```

  Cette ligne va forcer le serveur á dèmarrer en tant qu'utilisateur `mysql`, même si vous dèmarrez le serveur manuellement ou avec les scripts `safe_mysqld`, ou `mysql.server`. Pour plus de dètails, voyez Section A.3.2 [Changing MySQL user], page 696.

- N'acceptez pas les liens symboliques pour les tables (cette fonctionnalitè peut être dèsactivèe avec l'option `--skip-symlink` option). Il est gènèralement important si vous exècutez `mysqld` en tant que root, car tout le monde aurait accés en ècriture au dossier de donnèes de MySQL, et pourrait alors effacer des fichiers systémes. Voir Section 5.6.1.2 [Symbolic links to tables], page 402.

- Vèrifiez que l'utilisateur Unix qui exècute `mysqld` est le seul utilisateur avec les droits de lecture et ècriture dans le dossier de base de donnèes.

- Ne donnez pas le droit de `PROCESS` á tous les utilisateurs. La liste fournie par `mysqladmin processlist` affiche le texte des requêtes actuellement exècutèes, ce qui permet á toute personne pouvant exècuter cette commande de lire des valeurs qui seraient en clair, comme : `UPDATE user SET password=PASSWORD('not_secure')`.

  `mysqld` rèserve une connexion supplèmentaire pour les utilisateurs qui ont le droit de `PROCESS`, afin que le `root` MySQL puisse toujours se connecter et vèrifier que tout fonctionne bien, même s'il ne reste plus de connexions libres pour les autres utilisateurs.

- Ne donnez pas le droit de `FILE` á tous les utilisateurs. Tout utilisateur qui posséde ce droit peut ècrire un fichier n'importe oú sur le serveur, avec les droits hèritès du dèmon `mysqld` ! Pour rendre cela plus sècuritaire, tous les fichiers gènèrès par `SELECT ... INTO OUTFILE` sont lisibles par tous, mais personne ne peut les modifier.

  Le droit de `FILE` peut aussi être utilisè pour lire n'importe quel fichier accessible en lecture au dèmon qui fait tourner MySQL. Il devient donc possible, suivant les con-

figurations, d'utiliser la commande `LOAD DATA` sur le fichier '`/etc/passwd`' pour tout mettre en table, et ensuite le relire avec la commande `SELECT`.

- Si vous ne faites pas confiance á votre DNS, vos pouvez simplement utiliser des adresses IP au lieu des noms d'hôtes. Dans ce cas, soyez trés prudents lors de la crèation de droits qui utilisent des caractéres joker.
- Si vous voulez restreindre le nombre de connexions d'un utilisateur, vous pouvez le faire en utilisant la variable `max_user_connections` de `mysqld`.

4.2.3 Options de dèmarrage qui concernent la sècuritè

Les options suivantes de `mysqld` affectent la sècuritè :

`--local-infile[=(0|1)]`
: If one uses `--local-infile=0` then one can't use `LOAD DATA LOCAL INFILE`.

`--safe-show-database`
: Avec cette option, la commande `SHOW DATABASES` ne retourne que les bases pour lesquelles l'utilisateur courant a des droits. Depuis la verison 4.0.2, cette option est abandonnèe, et ne sert plus á rien (elle est activèe par dèfaut), car dèsormais, il y a le droit de `SHOW DATABASES`. Voir Section 4.3.1 [GRANT], page 226.

`--safe-user-create`
: Si cette option est activèe, tout utilisateur ne peut crèer d'autres utilisateurs avec les droits de `GRANT`, s'il ne dispose pas des droits d'insertion dans la table `mysql.user`. Si vous voulez donner un accés á un utilisateur pour qu'il puisse crèer des utilisateurs avec les droits dont il dispose, vous pouvez lui donner les droits suivants :

```
mysql> GRANT INSERT(user) ON mysql.user TO 'user'@'hostname';
```

  Cela va s'assurer que l'utilisateur ne peut par modifier une colonne directement, mais qu'il peut exècuter la commande `GRANT` sur d'autres utilisateurs.

`--skip-grant-tables`
: Cette option force le serveur á ne pas utiliser les tables de droits. Cette option donne donc tous les droits á tout le monde sur le serveur ! Vous pouvez forcer un serveur en fonctionnement á reprendre les tables de droits en exècutant la commande `mysqladmin flush-privileges` ou `mysqladmin reload`.)

`--skip-name-resolve`
: Les noms d'hôtes ne sont pas rèsolus. Toutes les valeurs de la colonne `Host` dans les tables de droits doivent être des adresses IP, ou bien `localhost`.

`--skip-networking`
: Ne pas accepter les connexions TCP/IP venant du rèseau. Toutes les connexions au serveur `mysqld` doivent être faîtes avec les sockets Unix. Cette option n'existe pas pour les versions antèrieures á la 3.23.27, avec les MIT-pthread, car les sockets Unix n'ètaient pas supportès par les MIT-pthreads á cette èpoque.

`--skip-show-database`
: Ne pas autoriser la commande `SHOW DATABASES`, á moins que l'utilisateur n'ait les droits de `SHOW DATABASES`. Depuis la version 4.0.2, vous n'avez plus besoin

de cette option, car les accés sont dèsormais donnès spècifiquement avec le droit `SHOW DATABASES`.

4.2.4 Problémes de sècuritè avec LOAD DATA LOCAL

Depuis MySQL 3.23.49 et MySQL 4.0.2, nous avons ajoutè de nouvelles options pour traiter les problémes de sècuritè liès á `LOAD DATA LOCAL`.

Il existe deux problémes particuliers pour le support de cette commande :

Comme la lecture du fichier est rèalisèe depuis le serveur, il est possible thèoriquement de crèer un serveur MySQL patchè qui pourrait lire n'importe quel fichier sur la machine cliente, qui serait accessible á l'utilisateur.

Dans un environnement web, oú les clients se connectent depuis un serveur web, un utilisateur peut se servir de la commande `LOAD DATA LOCAL` pour lire les fichiers qui sont sur le serveur web, et auquel ce dernier a accés (en supposant qu'un utilisateur peut exècuter n'importe quelle commande sur le serveur).

Il y a deux protections distinctes pour ces problémes :

Si vous ne configurez pas MySQL avec l'option `--enable-local-infile`, alors `LOAD DATA LOCAL` sera dèsactivè par tous les clients, á moins que l'option `mysql_options(... MYSQL_OPT_LOCAL_INFILE, 0)` soit activèe dans le client. Voir Section 8.4.3.38 [`mysql_options()`], page 641.

Pour le client en ligne de commande `mysql`, `LOAD DATA LOCAL` peut être activè en spècifiant l'option `--local-infile[=1]`, ou dèsactivè avec `--local-infile=0`.

Par dèfaut, tous les clients MySQL et les librairies sont compilès avec `--enable-local-infile`, pour être compatible avec MySQL 3.23.48 plus ancien.

Vous pouvez dèsactiver toutes les commandes `LOAD DATA LOCAL` du serveur MySQL en dèmarrant `mysqld` avec `--local-infile=0`.

Au cas oú `LOAD DATA LOCAL INFILE` est dèsactivè sur le serveur ou le client, vous obtiendrez le message d'erreur (1148) :

```
The used command is not allowed with this MySQL version
```

4.2.5 Rôle du systéme de priviléges

La fonction premiére du systéme de priviléges de MySQL est d'authentifier les utilisateurs se connectant á partir d'un hôte donnè, et de leur associer des priviléges sur une base de donnèes comme `SELECT`, `INSERT`, `UPDATE` et `DELETE`.

Les fonctionnalitès additionnelles permettent d'avoir un utilisateur anonyme et de contrôler les priviléges pour les fonctions spècifiques á MySQL comme `LOAD DATA INFILE` et les opèrations administratives.

4.2.6 Comment fonctionne le systéme de droits

Le systéme de droits de MySQL s'assure que les utilisateurs font exactement ce qu'ils sont supposès pouvoir faire dans la base. Lorsque vous vous connectez au serveur, vous identitè

est dèterminèe par **l'hôte d'oú vous vous connectez** et **le nom d'utilisateur que vous spècifiez**. Le systéme donne les droits en fonction de votre identitè et de **ce que vous voulez faire**.

MySQL considére votre nom d'hôte et d'utilisateur pour vous identifier, car il n'y pas que peu de raisons de supposer que le même nom d'utilisateur appartient á la même personne, quelque soit son point de connexion sur Internet. Par exemple, l'utilisateur `joe` qui se connecte depuis `office.com` n'est pas forcèment la même personne que `joe` qui se connecte depuis `elsewhere.com`. MySQL gére cela en vous aidant á distinguer les diffèrents utilisateurs et hôtes qui ont le même nom : vous pourriez donner des droits á `joe` lorsqu'il utilise sa connexion depuis `office.com`, et un autre jeu de droits lorsqu'il se connecte depuis `elsewhere.com`.

Le contrôle d'accés de MySQL se fait en deux ètapes :

- Etape 1 : Le serveur vèrifie que vous êtes autorisè á vous connecter.
- Etape 2 : En supposant que vous pouvez vous connecter, le serveur vèrifie chaque requête que vous soumettez, pour vèrifier si vous avez les droits suffisants pour l'exècuter. Par exemple, si vous sèlectionnez des droits dans une table, ou effacez une table, le serveur s'assure que vous avez les droits de `SELECT` pour cette table, ou les droits de `DROP`, respectivement.

Le serveur utilise les tables `user`, `db` et `host` dans la base `mysql` durant les deux ètapes. Les champs de cette table sont les suivants :

Nom de la table	`user`	`db`	`host`
Identifiant	`Host`	`Host`	`Host`
	`User`	`Db`	`Db`
	`Password`	`User`	
Champs de droits	`Select_priv`	`Select_priv`	`Select_priv`
	`Insert_priv`	`Insert_priv`	`Insert_priv`
	`Update_priv`	`Update_priv`	`Update_priv`
	`Delete_priv`	`Delete_priv`	`Delete_priv`
	`Index_priv`	`Index_priv`	`Index_priv`
	`Alter_priv`	`Alter_priv`	`Alter_priv`
	`Create_priv`	`Create_priv`	`Create_priv`
	`Drop_priv`	`Drop_priv`	`Drop_priv`
	`Grant_priv`	`Grant_priv`	`Grant_priv`
	`References_priv`		
	`Reload_priv`		
	`Shutdown_priv`		
	`Process_priv`		
	`File_priv`		

Lors de la seconde ètape du contrôle d'accés (vèrification de la requête), le serveur peut, suivant la requête, consulter aussi les tables `tables_priv` et `columns_priv`. Les champs de ces tables sont :

Nom de la table	`tables_priv`	`columns_priv`
Identifiant	`Host`	`Host`

	Db	Db
	User	User
	Table_name	Table_name
		Column_name
Champs de droits	Table_priv	Column_priv
	Column_priv	
Autre champs	Timestamp	Timestamp
	Grantor	

Chaque table de droit contient des champs d'identification et des champs de droits.

Les champs d'identification dèterminent quels utilisateurs correspondent á cette ligne dans la table. Par exemple, une ligne dans la table `user` avec les valeurs dans les colonnes `Host` et `User` de `'thomas.loc.gov'` et `'bob'` servira á identifier les connexions qui sont faites par l'utilisateur `bob` depuis l'hôte `thomas.loc.gov`. De même, une ligne dans la table `db` avec les valeurs des colonnes `Host`, `User` et `Db` de `'thomas.loc.gov'`, `'bob'` et `'reports'` sera utilisèe lorsque l'utilisateur `bob` se connecte depuis l'hôte `thomas.loc.gov` pour accèder á la base `reports`. Les tables `tables_priv` et `columns_priv` contiennent en plus des champs indiquant les tables et combinaisons tables et colonnes auxquelles les lignes s'appliquent.

Pour les contrôles d'accés, les comparaisons de nom d'hôte avec la colonne `Host` sont insensibles á la casse. Les colonnes `User`, `Password`, `Db` et `Table_name` sont sensibles á la casse. Les valeurs de la colonne `Column_name` sont insensibles á la casse pour les versions de MySQL 3.22.12 et plus rècent.

Les champs de droits indiquent si le droit est donnè, c'est á dire si l'opèration indiquèe peut être exècutè. Le serveur combine les informations dans diffèrentes tables pour former une description compléte de l'utilisateur. Les régles utilisées sont dècrites dans Section 4.2.10 [Request access], page 219.

Les champs d'identification sont des chaînes, dèclarèes comme suit. La valeur par dèfaut de chacun des champs est la chaîne vide.

Nom du champs	Type	Notes
`Host`	`CHAR(60)`	
`User`	`CHAR(16)`	
`Password`	`CHAR(16)`	
`Db`	`CHAR(64)`	(`CHAR(60)` pour les tables `tables_priv` et `columns_priv`)
`Table_name`	`CHAR(60)`	
`Column_name`	`CHAR(60)`	

Dans les tables `user`, `db` et `host`, tous les champs de droits sont dèclarès avec le type `ENUM('N','Y')` : il peuvent prendre tous les valeurs de `'N'` (non) ou `'Y'` (oui, YES), et la valeur par dèfaut est `'N'`.

Dans les tables `tables_priv` et `columns_priv`, les champs de droits sont dèclarès comme des champs de type `SET` :

Nom de la table	Nom du champs	Valeurs possibles
`tables_priv`	`Table_priv`	`'Select', 'Insert', 'Update', 'Delete', 'Create', 'Drop', 'Grant', 'References', 'Index', 'Alter'`

```
tables_priv   Column_     'Select', 'Insert', 'Update',
              priv        'References'
columns_      Column_     'Select', 'Insert', 'Update',
priv          priv        'References'
```

En bref, le serveur utilise les tables de droits comme ceci :

- La table `user` dètermine si le serveur accepte ou rejette la connexion. Pour les connexions acceptèes, tous les priviléges donnès dans la table `user` indiquent des priviléges globaux. Ces droits d'appliquent á **toutes** les bases du serveur.
- Les tables `db` et `host` sont utilisèes conjointement :
  - Les champs d'identification de la table `db` dèterminent quels utilisateurs peuvent accèder á quelles bases, depuis quel hôte. Les champs de droits indiquent alors les opèrations permises.
  - La table `host` est utilisèe comme extension de la table `db` lorsque vous voulez qu'une ligne de la table `db` s'applique á plusieurs hôtes. Par exemple, si vous voulez qu'un utilisateur soit capable d'utiliser une base depuis plusieurs hôtes dans votre rèseau, laissez la colonne `Host` vide dans la table `db`, Ce mècanisme est dècrit en dètails dans Section 4.2.10 [Request access], page 219.
- Les tables `tables_priv` et `columns_priv` sont similaires á la table `db`, mais sont plus atomiques : elle s'appliquent au niveau des tables et des colonnes, plutôt qu'au niveau des bases.

Notez que les droits d'administration tels que (`RELOAD`, `SHUTDOWN`, etc...) ne sont spècifiès que dans la table `user`. En effet, ces opèrations sont des opèrations au niveau serveur, et ne sont pas liées á une base de donnèes, ce qui fait qu'il n'y a pas de raison de les lier avec les autres tables. En fait `user` doit être consultè pour dèterminer les autorisations d'administration.

Le droit de `FILE` est spècifiè par la table `user`. Ce n'est pas un droit d'administration, mais votre capacitè á lire ou ècrire des fichiers sur le serveur hôte et dèpendant de la base á laquelle vous accèdez.

Le serveur `mysqld` lit le contenu des tables de droits une fois, au dèmarrage. Lorsqu'il y a des modifications dans les tables, elles prennent effet tel qu'indiquè dans Section 4.3.3 [Privilege changes], page 232.

Lorsque vous modifiez le contenu des tables de droits, c'est une bonne idèe que de s'assurer que vous avez bien configurè les droits qui vous intèressent. Pour vous aider dans votre diagnostique, voyez Section 4.2.11 [Access denied], page 222. Pour des conseils sur les aspects sècuritè, voyez Section 4.2.2 [Security], page 206.

Un outil de diagnostique pratique est le script `mysqlaccess`, que Yves Carlier a fourni á la distribution MySQL. Appelez `mysqlaccess` avec l'option the `--help` pour comprendre comment il fonctionne. Notez que `mysqlaccess` ne vèrifie les accés que pour les tables `user`, `db` et `host`. Il n'utilise pas les tables de droit de niveau table ou colonne.

4.2.7 Droits fournis par MySQL

Les droits des utilisateurs sont stockès dans les tables `user`, `db`, `host`, `tables_priv` et `columns_priv` de la base `mysql` (c'est-á-dire, la base nommèe `mysql`). Le serveur MySQL

lit ces tables au dèmarrage, et dans les circonstances indiquèes dans la section Section 4.3.3 [Privilege changes], page 232.

Les noms utilisès dans ce manuel font rèfèrence aux droits fournis par MySQL version 4.0.2, tel que prèsentès dans la table ci-dessous, avec le nom de la colonne associèe au droit, dans la table de droits, et dans le contexte d'application :

Droit	**Colonne**	**Contexte**
`ALTER`	`Alter_priv`	tables
`DELETE`	`Delete_priv`	tables
`INDEX`	`Index_priv`	tables
`INSERT`	`Insert_priv`	tables
`SELECT`	`Select_priv`	tables
`UPDATE`	`Update_priv`	tables
`CREATE`	`Create_priv`	bases de donnèes, tables ou index
`DROP`	`Drop_priv`	bases de donnèes ou tables
`GRANT`	`Grant_priv`	bases de donnèes ou tables
`REFERENCES`	`References_priv`	bases de donnèes ou tables
`CREATE TEMPORARY TABLES`	`Create_tmp_table_priv`	administration du serveur
`EXECUTE`	`Execute_priv`	administration du serveur
`FILE`	`File_priv`	accés aux fichiers du serveur
`LOCK TABLES`	`Lock_tables_priv`	administration du serveur
`PROCESS`	`Process_priv`	administration du serveur
`RELOAD`	`Reload_priv`	administration du serveur
`REPLICATION CLIENT`	`Repl_client_priv`	administration du serveur
`REPLICATION SLAVE`	`Repl_slave_priv`	administration du serveur
`SHOW DATABASES`	`Show_db_priv`	administration du serveur
`SHUTDOWN`	`Shutdown_priv`	administration du serveur
`SUPER`	`Super_priv`	administration du serveur

Les droits de `SELECT`, `INSERT`, `UPDATE` et `DELETE` vous permettent de faire des opèrations sur les lignes qui existent, dans une table existante d'une base.

La commande `SELECT` requiert le droit de `SELECT` uniquement si des lignes sont lues dans une une table. Vous pouvez exèctuer une commande `SELECT` même sans aucun droit d'accés á une base de donnèes dans le serveur. Par exemple, vous pourriez utiliser le client `mysql` comme une simple calculatrice :

```
mysql> SELECT 1+1;
mysql> SELECT PI()*2;
```

Le droit de `INDEX` vous donne le droit de crèer et dètruire des index de table.

Le droit de `ALTER` vous donne le droit de modifier une table avec la commande `ALTER TABLE`.

Les droits de `CREATE` et `DROP` vous permettent de crèer de nouvelles tables et bases de donnèes, et de les supprimer.

Notez que si vous donnez le droit de `DROP` pour la base de donnèes `mysql` á un utilisateur, cet utilisateur pourra dètruire la base qui contient les droits d'accés du serveur !

Le droit de `GRANT` vous permet de donner les droits que vous possèdez á d'autres utilisateurs.

Le droit de `FILE` vous donne la possibilitè de lire et ècrire des fichiers sur le serveur avec les commandes `LOAD DATA INFILE` et `SELECT ... INTO OUTFILE`. Tout utilisateur qui posséde ce droit peut donc lire ou ècrire dans n'importe quel fichier á l'intèrieur duquel le serveur MySQL peut lire ou ècrire.

Les autres droits sont utilisès pour les opèrations administratives qui sont exècutèes par l'utilitaire `mysqladmin`. La table ci-dessous montre quelle commande est associèe á `mysqladmin` avec un de ces droits :

Droit	**Commande autorisèe**
`RELOAD`	`reload`, `refresh`, `flush-privileges`, `flush-hosts`, `flush-logs` et `flush-tables`
`SHUTDOWN`	`shutdown`
`PROCESS`	`processlist`
`SUPER`	`kill`

La commande `reload` indique au serveur de relire les tables de droits. La commande `refresh` vide les tables de la mèmoire, ècrit les donnèes et ferme le fichier de log. `flush-privileges` est un synonyme de `reload`. Les autres commandes `flush-*` effectuent des fonctions similaires á la commande `refresh` mais sont plus limitèes dans leur application, et sont prèfèrables dans certains contextes. Par exemple, si vous souhaitez simplement vider les tampons dans le fichier de log, utilisez `flush-logs`, qui est un meilleur choix que `refresh`.

La commande `shutdown` èteint le serveur.

La commande `processlist` affiche les informations sur les threads qui s'exècutent sur le serveur. La commande `kill` termine un des threads du serveur. Vous pouvez toujours afficher et terminer vos propres threads, mais vous aurez besoin des droits de `PROCESS` pour afficher les threads, et le droit de `SUPER` pour terminer ceux qui ont ètè dèmarrès par d'autres utilisateurs. Voir Section 4.5.5 [KILL], page 266.

C'est une bonne idèe en gènèral, de ne donner les droits de Grant qu'aux utilisateurs qui en ont besoin, et vous devriez être particuliérement vigilant pour donner certains droits :

- Le droit de `GRANT` permet aux utilisateurs de donner leurs droits á d'autres utilisateurs. Deux utilisateurs avec des droits diffèrents et celui de `GRANT` pourront combiner leurs droits respectifs pour gagner un autre niveau d'utilisation du serveur.
- Le droit de `ALTER` peut être utilisè pour tromper le systéme en renommant les tables.
- Le droit de `FILE` peut servir á lire des fichiers accessibles á tous sur le serveur, et les placer dans une base de donnèes. Le contenu pourra alors être lu et manipulè avec `SELECT`. Cela inclus le contenu de toutes les bases actuellement hèbergèes sur le serveur !
- Le droit de `SHUTDOWN` peut conduire au dènis de service, en arrêtant le serveur.
- Le droit de `PROCESS` permet de voir en texte clair les commandes qui s'exècutent actuellement, et notamment les changements de mot de passe.

- Les droits sur la base de donnèes `mysql` peuvent être utilisès pour changer des mots de passe ou des droits dans la table des droits (Les mots de passe sont stockès chiffrès, ce qui èvite que les intrus ne les lisent). S'ils accédent á un mot de passe dans la table `mysql.user`, ils pourront l'utiliser pour se connecter au serveur avec cet utilisateur (avec des droits suffisants, le même utilisateur pourra alors remplacer un mot de passe par un autre).

Il y a des choses qui ne sont pas possibles avec le systéme de droits de MySQL :

- Vous ne pouvez pas explicitement interdire l'accés á un utilisateur spècifique. C'est á dire, vous ne pouvez pas explicitement dècrire un utilisateur et lui refuser la connexion.
- Vous ne pouvez pas spècifier qu'un utilisateur a les droits de crèer et de supprimer des tables dans une base, mais qu'il n'a pas les droits pour crèer et supprimer cette base.

4.2.8 Se connecter au serveur MySQL

Les clients MySQL requiérent gènèralement que vous spècifiez les paramétres de connexion pour vous connecter au serveur MySQL : l'hôte que vous voulez utiliser, votre nom d'utilisateur et votre mot de passe. Par exemple, le client `mysql` peut être dèmarrè comme ceci (les arguments optionnels sont entre crochets '`[`' et '`]`') :

```
shell> mysql [-h nom_d_hote] [-u nom_d_utilisateur] [-pvotre_mot_de_passe]
```

Les formes alternatives des options `-h`, `-u`, and `-p` sont `--host=host_name`, `--user=user_name` et `--password=your_pass`. Notez qu'il n'y a *aucun espace* entre l'option `-p` ou `--password=` et le mot de passe qui le suit.

Note : spècifier un mot de passe en ligne de commande n'est pas sècuritaire ! Tout utilisateur de votre serveur peut dècouvrir votre mot de passe en tapant la commande : `ps auxww`. Voir Section 4.1.2 [Option files], page 198.

`mysql` utilise des valeurs par dèfaut pour chacun des paramétes qui manquent en ligne de commande :

- Le nom d'hôte par dèfaut est `localhost`.
- Le nom d'utilisateur par dèfaut est votre nom d'utilisateur de systéme Unix.
- Aucun mot de passe n'est transmis si `-p` manque.

Par exemple, pour un utilisateur Unix `joe`, les commandes suivantes sont èquivalentes :

```
shell> mysql -h localhost -u joe
shell> mysql -h localhost
shell> mysql -u joe
shell> mysql
```

Les autres clients MySQL se comportent de maniére similaire.

Sous Unix, vous pouvez spècifier diffèrentes valeurs par dèfaut qui seront utilisèes lorsque vous ètablierez la connexion, de maniére á ce que vous n'ayez pas á entrer ces informations en ligne de commande lorsque vous invoquez un programme client. Cela peut se faire de plusieurs façons :

- Vous pouvez spècifier les informations de connexion dans la section `[client]` du fichier de configuration '`.my.cnf`' de votre dossier personnel. La section qui vous interesse ressemble á ceci :

```
[client]
host=nom_d_hote
user=nom_d'utilisateur
password=votre_mot_de_passe
```

Voir Section 4.1.2 [Option files], page 198.

- Vous pouvez spècifier les paramétres de connexion avec les variables d'environnement. L'hôte peut être spècifiè á `mysql` avec la variable `MYSQL_HOST`. L'utilisateur MySQL peut être spècifiè avec la variable `USER` (uniquement pour Windows). Le mot de passe peut être spècifiè avec `MYSQL_PWD` : mais ceci est peu sècuritaire. Voir la prochaine section Voir Annexe F [Environment variables], page 828.

4.2.9 Contrôle d'accés, ètape 1 : Vèrification de la connexion

Lorsque vous tentez de vous connecter au serveur MySQL, le serveur accepte ou rejette la connexion en fonction de votre identitè et du mot de passe que vous fournissez. Si le mot de passe ne correspond pas á celui qui est en base, le serveur vous interdit complétement l'accés. Sinon, le serveur accepte votre connexion et passe á l'ètape 2, et la gestion de commandes.

Votre identitè est basèe sur trois informations :

- L'hôte depuis lequel vous vous connectez
- Votre nom d'utilisateur MySQL
- Votre mot de passe

La vèrification d'identitè est rèalisèe avec les trois colonnes de la table `user` (`Host`, `User` et `Password`). Le serveur accepte la connexion uniquement si une entrèe dans la table `user` correspond á votre hôte, et que vous fournissez le mot de passe qui correspond.

Les valeurs de la table `user` peuvent être paramètrèes comme ceci :

- Une valeur de la colonne `Host` peut être un nom d'hôte, une adresse IP numèrique, ou encore `'localhost'`, qui reprèsente l'hôte local.
- Vous pouvez utiliser les caractéres jokers '`%`' et '`_`' dans le champ `Host`.
- Une valeur de `'%'` dans la colonne `Host` remplace n'importe quelle autre valeur.
- Une valeur vide pour la colonne `Host` indique que les droits doivent être gèrès avec les entrèes de la table `host` qui correspond á l'hôte se connectant. Vous trouverez plus d'informations á ce sujet dans le prochain chapitre.
- Depuis MySQL version 3.23, les valeurs de `Host` spècifièes sous la forme d'IP numèriques peuvent être complètèes avec le masque de rèseau qui indique combien de bits d'adresse sont utilisès. Par exemple :

  ```
  mysql> GRANT ALL PRIVILEGES ON db.*
      -> TO david@'192.58.197.0/255.255.255.0';
  ```

  Cela permet á toute personne se connectant depuis une adresse IP qui satisfait la contrainte suivante :

  ```
  user_ip & netmask = host_ip.
  ```

  Dans l'exemple ci-dessus, toutes les IP dans l'intervalle 192.58.197.0 - 192.58.197.255 pourront se connecter au serveur MySQL.

- Les caractéres joker ne sont pas autorisès dans le champ `User`, mais vous pouvez spècifier une valeur vide qui remplacera tous les noms. Si l'entrèe de la table `user`, qui correspond á une connexion entrante, a un nom d'utilisateur vide, l'utilisateur est alors considèrè comme anonyme (c'est l'utilisateur sans nom) et non pas le nom d'utilisateur fourni. Cela signifie qu'un utilisateur fournissant un nom d'utilisateur vide sera utilisè pour identifier ses droits dans les prochaines ètapes.
- Le champ `Password` peut être vide. Cela ne signifie pas que n'importe quel mot de passe est valable, mais que l'utilisateur peut se connecter sans fournir de mot de passe.

Les valeurs non vides du champ `Password` reprèsentent des valeurs du mot de passe chiffrèes. MySQL ne stocke pas les mots de passe en clair, á la vue de tous. Au contraire, le mot de passe fourni pas l'utilisateur qui tente de se connecter est chiffrè (avec la fonction `PASSWORD()`). Le mot de passe ainsi chiffrè est alors utilisè entre le client et le serveur pour vèrifier s'il est valable. Cela èvite que des mots de passe en clair circulent entre le client et le serveur, sur la connexion. Notez que du point de vue de MySQL, le mot de passe chiffrè est le vrai mot de passe, ce qui fait que vous ne devez en aucun cas le donner á un tiers. En particulier, ne donnez pas accés en lecture aux utilisateurs normaux aux tables d'administration dans la base `mysql`!

Les exemples ci-dessous illustrent comment diffèrentes variantes de `Host` et `User` dans la table `user` s'appliquent aux connexion entrantes :

`Host` value	`User` value	Connexions autorisèes
`'thomas.loc.gov'`	`'fred'`	`fred`, se connectant depuis `thomas.loc.gov`
`'thomas.loc.gov'`	`''`	N'importe quel utilisateur, se connectant depuis `thomas.loc.gov`
`'%'`	`'fred'`	`fred`, se connectant depuis n'importe quel hôte
`'%'`	`''`	N'importe quel utilisateur, se connectant depuis n'importe quel hôte
`'%.loc.gov'`	`'fred'`	`fred`, se connectant depuis n'importe quel hôte dans le domaine `loc.gov`
`'x.y.%'`	`'fred'`	`fred`, se connectant depuis `x.y.net`, `x.y.com`,`x.y.edu`, etc. (Ceci n'est probablement pas trés utilisè)
`'144.155.166.177'`	`'fred'`	`fred`, se connectant depuis l'hôte d'IP `144.155.166.177`
`'144.155.166.%'`	`'fred'`	`fred`, se connectant depuis un hôte d'IP dans la classe C `144.155.166`
`'144.155.166.0/255.255.255.0'`	`'fred'`	Identique á l'exemple prècèdent

Comme vous pouvez utiliser des caractéres jokers dans les adresses IP de la colonne `Host` (par exemple, `'144.155.166.%'` pour identifier tout un sous-rèseau), il est possible d'exploiter cette fonctionnalitè en nommant un hôte `144.155.166.kekpart.com`. Pour contrer de telles tentatives, MySQL interdit les caractéres jokers avec les noms d'hôtes qui commencent par des chiffres ou des points. Par exemple, si vous avez un nom d'hôte tel que `1.2.foo.com`, il ne sera jamais trouvè dans la colonne `Host` des tables de droits. Seule une adresse IP numèrique peut être comparèe avec un masque á caractére joker.

Une connexion entrante peut être identifiée par plusieurs entrèes dans la table `user`. Par exemple, une connexion depuis `thomas.loc.gov` par `fred` sera identifiée de plusieurs façons

diffèrentes dans l'exemple ci-dessus. Comment le serveur va-t-il choisir si il y a plusieurs solutions ? Le serveur rèsoud cette question en triant les utilisateurs dans la colonne `user` aprés le dèmarrage, et il recherchera dans cette liste triée lorsqu'un utilisateur tente de se connecter. La premiére ligne qui satisfait les critéres sera alors utilisée.

Le tri de la table `user` suit les régles suivantes. Supposons que votre table `user` ressemble á ceci :

```
+-----------+----------+-
| Host      | User     | ...
+-----------+----------+-
| %         | root     | ...
| %         | jeffrey  | ...
| localhost | root     | ...
| localhost |          | ...
+-----------+----------+-
```

Lorsque le serveur lit cette table, il ordonne les lignes depuis les valeurs les plus spècialisées de la colonne `Host` jusqu'aux plus gènèrales (`'%'` dans la colonne `Host` signifie "tous les hôtes" et elle est la moins spècifique). Les entrèes identiques dans la colonne `Host` sont ordonnèes en fonction de la spècificitè des valeurs de la colonne `User` (une entrèe vide dans la colonne `User` signifie "n'importe quel utilisateur" et est spècifique). Le rèsultat de ce tri donne quelque chose comme ceci :

```
+-----------+----------+-
| Host      | User     | ...
+-----------+----------+-
| localhost | root     | ...
| localhost |          | ...
| %         | jeffrey  | ...
| %         | root     | ...
+-----------+----------+-
```

Lorsqu'une connexion est en cours de mise en place, le serveur regarde dans cette liste, et utilisera la premiére entrèe trouvèe. Pour une connexion depuis l'hôte `localhost` avec le nom d'utilisateur `jeffrey`, les entrèes `'localhost'` dans la colonne `Host` sont trouvèes en premier. Parmi celles-la, la ligne avec un utilisateur vide satisfait les deux contraintes sur le nom et l'hôte. `'%'/'jeffrey'` pourrait avoir fonctionnè, mais comme ce n'est pas le premier rencontrè, il n'est pas utilisè.

Voici un autre exemple. Supposons que la table `user` ressemble á ceci :

```
+----------------+----------+-
| Host           | User     | ...
+----------------+----------+-
| %              | jeffrey  | ...
| thomas.loc.gov |          | ...
+----------------+----------+-
```

La table triée ressemble á ceci :

```
+----------------+----------+-
| Host           | User     | ...
+----------------+----------+-
```

```
| thomas.loc.gov |          | ...
| %              | jeffrey  | ...
+----------------+----------+-
```

Une connexion depuis l'hôte `thomas.loc.gov` avec `jeffrey` satisfait les conditions de la premiére ligne, tandis qu'une connexion depuis `whitehouse.gov` avec `jeffrey` satisfait la seconde ligne.

Une erreur commune est de penser que pour un utilisateur donnè, toutes les entrèes qui utilisent explicitement ce nom seront utilisèes en premier lorsque la connexion est en cours d'ètablissement. Ceci est tout simplement faux. L'exemple prècèdent illustre cette situation, car la connexion depuis l'hôte `thomas.loc.gov` avec `jeffrey` est la premiére ligne qui est trouvèe, alors que la ligne contenant `'jeffrey'` dans la colonne `User` est ignorèe, car il n'y a pas de nom d'utilisateur.

Si vous avez des problémes lors de la connexion, listez le contenu de la table `user` et triez-la á la main pour savoir quel est la premiére ligne qui est trouvèe et utilisèe.

4.2.10 Contrôle d'accés, ètape 2 : Vèrification de la requête

Une fois que vous avez ètabli la connexion, le serveur passe á l'ètape 2. Pour chaque requête qui est fournie avec la connexion, le serveur vèrifie si vous avez les droits suffisants pour exècuter une commande, en fonction du type de commande. C'est á ce moment que les colonnes de droits des tables d'administration entrent en scéne. Ces droits peuvent provenir de la table `user`, `db`, `host`, `tables_priv` ou `columns_priv`. Les tables d'administration sont manipulèes avec les commandes `GRANT` et `REVOKE`. Voir Section 4.3.1 [GRANT], page 226. (Vous pouvez aussi vous reporter á la section Section 4.2.6 [Privileges], page 209 qui liste les champs prèsents dans chaque table d'administration).

La table d'administration `user` donne les droits aux utilisateurs au niveau global, c'est á dire que ces droits s'appliquent quelle que soit la base de donnèes courante. Par exemple, si la table `user` vous donne le droit d'effacement ,`DELETE`, vous pouvez effacer des donnèes dans n'importe quelle base de ce serveur. En d'autres termes, les droits stockès dans la table `user` sont des droits de super utilisateur. Il est recommandè de ne donner des droits via la table `user` uniquement aux super utilisateurs, ou aux administrateurs de bases. Pour les autres utilisateurs, il vaut mieux laisser les droits dans la table `user` á `'N'` et donner des droits au niveau des bases uniquement, avec les tables `db` et `host`.

Les tables `db` et `host` donnent des droits au niveau des bases. Les droits peuvent être spècifiès dans ces tables comme ceci :

- Les caractéres '`%`' et '`_`' peuvent être utilisès dans la colonne `Host` et `Db` des deux tables. Si vous souhaitez utiliser le caractére '`_`' comme nom de base, utiliser la sèquence '`\_`' dans la commande `GRANT`.
- La valeur `'%'` dans la colonne `Host` de la table `db` signifie "tous les hôtes". Une valeur vide dans la colonne `Host` de la table `db` signifie "consulte la table `host` pour plus de dètails".
- La valeur `'%'` ou vide dans la colonne `Host` de la table `host` signifie "tous les hôtes".
- La valeur `'%'` ou vide dans la colonne `Db` des deux tables signifie "toutes les bases de donnèes".

- Un utilisateur vide dans la colonne `User` de l'une des deux tables identifie l'utilisateur anonyme.

Les tables `db` et `host` sont lues et triées par le serveur au dèmarrage (en même temps que la table `user`. La table `db` est triée suivant les valeurs des colonnes `Host`, `Db` et `User`, et la table `host` est triée en fonction des valeurs des colonnes `Host` et `Db`. Comme pour la table `user`, le tri place les entrèes les plus spècifiques au dèbut, et les plus gènèrales á la fin. Lorsque le serveur recherche une ligne, il utilise la premiére qu'il trouve.

Les tables `tables_priv` et `columns_priv` spècifient les droits au niveau des tables et des colonnes. Les valeurs des droits dans ces tables peuvent être spècifiès avec les caractéres spèciaux suivants :

- Les caractéres '`%`' et '`_`' peuvent être utilisès dans la colonne `Host` et `Db` des deux tables.
- La valeur `'%'` dans la colonne `Host` des deux tables signifie "tous les hôtes".
- Les colonnes `Db`, `Table_name` et `Column_name` ne peuvent pas contenir de valeur vide ou de caractéres jokers, dans les deux tables.

Les tables `tables_priv` et `columns_priv` sont triées en fonction des colonnes `Host`, `Db` et `User`. Ce tri est similaire á celui du tri de la table `db`, même si le tri est bien plus simple, car seul le champ `Host` peut contenir des caractéres jokers.

Le processus de vèrification est dècrit ci-dessous. Si vous êtes familier avec le code source de contrôle d'accés, vous noterez que la description différe lègérement de l'algorithme utilisè. La description est èquivalente á ce que fait en rèalitè le code. La diffèrence permet une meilleure approche pèdagogique.

Pour les requêtes d'administration comme `SHUTDOWN`, `RELOAD`, etc., le serveur vèrifie uniquement l'entrèe dans la table `user`, car c'est la seule table qui spècifie des droits d'administration. Le droit est donnè si la ligne utilisèe dans la connexion courante dans la table `user` donne le droit, et sinon, ce droit est interdit. Par exemple, si vous souhaitez exècuter la commande `mysqladmin shutdown` mais que votre ligne dans la table `user` ne vous en donne pas le droit (`SHUTDOWN`), vous n'aurez pas le droit sans même vèrifier les tables `db` ou `host` : ces tables ne contiennent pas de colonne `Shutdown_priv`, ce qui èvite qu'on en ait besoin.

Pour les requêtes exploitant une base de donnèes, comme `INSERT`, `UPDATE`, etc., le serveur vèrifie d'abord les droits globaux de l'utilisateur (droits de super utilisateur), en regardant dans la table `user`. Si la ligne utilisèe dans cette table donne droit á cette opèration, le droit est donnè. Si les droits globaux dans `user` sont insuffisants, le serveur dèterminera les droits spècifiques á la base avec les tables `db` et `host` :

1. Le serveur recherche dans la table `db` des informations en se basant sur les colonnes `Host`, `Db` et `User`. Les champs `Host` et `User` sont comparès avec les valeurs de l'hôte et de l'utilisateur qui sont connectès. Le champ `Db` est comparè avec le nom de la base de donnèes que l'utilisateur souhaite utiliser. S'il n'existe pas de ligne qui corresponde á `Host` et `User`, l'accés est interdit.
2. S'il existe une ligne dans la table `db` et que la valeur de la colonne `Host` n'est pas vide, cette ligne dèfinit les droits de l'utilisateur.
3. Si dans la ligne de la table `db`, la colonne `Host` est vide, cela signifie que la table `host` spècifie quels hôtes doivent être autorisès dans la base. Dans ce cas, une autre recherche est faite dans la table `host` pour trouver une ligne avec les colonnes `Host` et `Db`. Si

aucune ligne de la table `host` n'est trouvèe, l'accés est interdit. S'il y a une ligne, les droits de l'utilisateur sont calculès comme l'intersection (**NON PAS** l'union !) des droits dans les tables `db` et `host`, c'est-á-dire que les droits doivent être marquès `'Y'` dans les deux tables (de cette façon, vous pouvez donner des droits gènèraux dans la table `db` puis les restreindre sèlectivement en fonction des hôtes, en utilisant la table `host`.

Aprés avoir dèterminè les droits spècifiques á l'utilisateur pour une base grâce aux tables `db` et `host`, le serveur les ajoute aux droits globaux, donnès par la table `user`. Si le rèsultat autorise la commande demandèe, l'accés est donnè. Sinon, le serveur vèrifie les droits au niveau de la table et de la colonne dans les tables `tables_priv` et `columns_priv`, et les ajoute aux droits dèjá acquis. Les droits sont alors donnès ou rèvoquès en fonction de ces rèsultats.

Exprimèe en termes boolèens, la description prècèdente du calcul des droits peut être rèsumè comme ceci :

```
droits globaux
OR (droits de base AND droits d'hôte)
OR droits de table
OR droits de colonne
```

Il n'est peut-être pas èvident pourquoi, si les droits globaux issus de la table `user` sont initialement insuffisants pour l'opèration demandèe, le serveur ajoute ces droits á ceux de base, table ou colonne ? La raison est que la requête peut demander l'application de plusieurs droits. Par exemple, si vous exècutez une commande `INSERT ... SELECT`, vous aurez besoin des droits de `INSERT` et de `SELECT`. Vos droits peuvent être tels que la table `user` donne un droit, mais que la table `db` en donne un autre. Dans ce cas, vous aurez les droits nècessaires pour faire une opèration, mais le serveur ne peut le dèduire d'une seule table : les droits de plusieurs tables doivent être combinès pour arriver á la bonne conclusion.

La table `host` sert á gèrer une liste d'hôtes reconnus et sècuritaires.

Chez TcX, la table `host` contient une liste de toutes les machines du rèseau local. Ces machines reçoivent tous les droits.

Vous pouvez aussi utiliser la table `host` pour spècifier les hôtes qui **ne sont pas** sècuritaires. Supposons que la machine `public.votre.domaine` t est placèe dans une zone publique que vous considèrez comme peu sûre. Vous pouvez autoriser l'accés de toutes les machines, hormis celle-ci, grâce á la table `host` configurèe comme ceci :

```
+--------------------+----+-
| Host               | Db | ...
+--------------------+----+-
| public.your.domain | %  | ... (tous les droits á 'N')
| %.your.domain      | %  | ... (tous les droits á 'Y')
+--------------------+----+-
```

Naturellement, vous devriez toujours tester vos requêtes dans la table de droits, en utilisant l'utilitaire `mysqlaccess` pour vous assurer que vous disposez des droits nècessaires pour rèaliser cette opèration.

4.2.11 Causes des erreurs `Access denied`

Si vous rencontrez des erreurs `Access denied` quand vous essayez de vous connecter au serveur MySQL, la liste suivante indique quelques actions á entreprendre pour corriger le probléme :

- Aprés l'installation de MySQL, avez vous exècutè le script `mysql_install_db` pour initialiser le contenu de la table des droits ? Si tel n'est pas le cas, faites le. Voir Section 4.3.4 [Default privileges], page 232. Testez les priviléges initiaux en exècutant la commande suivante :

  ```
  shell> mysql -u root test
  ```

  Le serveur devrait vous laisser vous connecter sans erreurs. Vous devez aussi vous assurer que vous avez un fichier '`user.MYD`' dans le dossier des bases de donnèes MySQL. Ordinairement, il s'agit de '`CHEMIN/var/mysql/user.MYD`', oú `CHEMIN` est le chemin vers le dossier racine de l'installation MySQL.

- Aprés une installation toute fraîche, vous devez vous connecter au serveur et crèer les utilisateurs en règlant leurs permissions d'accés :

  ```
  shell> mysql -u root mysql
  ```

  Le serveur devrait vous laisser vous connecter car l'utilisateur `root` de MySQL n'a pas de mot de passe initial. Ceci est aussi une faille de sècuritè, et donc, vous devez choisir un mot de passe pour l'utilisateur `root` en même tant que les autres utilisateurs MySQL.

  Si vous essayez de vous connecter en `root` et que vous rencontrez cette erreur :

  ```
  Access denied for user: '@unknown' to database mysql
  ```

  cela signifie que vous n'avez pas d'entrèe dans la table `user` avec une valeur `'root'` de la colonne `User` et que `mysqld` cne peut trouver de nom d'hôte á votre client. Dans ce cas, vous devez redèmarrer le serveur avec l'option `--skip-grant-tables` et èditer le fichier '`/etc/hosts`' ou '`\windows\hosts`' pour ajouter une entrèe pour votre hôte.

- Si vous obtenez une erreur qui ressemble á celle-ci :

  ```
  shell> mysqladmin -u root -pxxxx ver
  Access denied for user: 'root@localhost' (Using password: YES)
  ```

  Cela signifie que vous utilisez un mot de passe erronè. Voir Section 4.3.7 [Passwords], page 237.

  Si vous avez oubliè le mot de passe root, vous pouvez redèmarrer `mysqld` avec `--skip-grant-tables` pour changer le mot de passe. Voir Section A.4.2 [Resetting permissions], page 700.

  Si vous obtenez l'erreur suivante même sans spècifier de mot de passe, cela signifie que vous avez un mauvais mot de passe dans un fichier `my.ini`. Voir Section 4.1.2 [Option files], page 198. Vous pouvez ne pas utiliser les fichiers d'options en utilisant l'option `--no-defaults`, comme suit :

  ```
  shell> mysqladmin --no-defaults -u root ver
  ```

- Si vous mettez á jour une installation MySQL existante á partir d'une version plus ancienne que 3.22.11 vers la version 3.22.11 ou plus á jour, avez-vous exècutè le script `mysql_fix_privilege_tables` ? Si ce n'est pas le cas, faites le. La structure des tables

de droits a changè avec la version 3.22.11 de MySQL lorsque la commande `GRANT` est devenue fonctionnelle.

- Si vos droits semblent avoir changè au milieu d'une session, il se peut qu'un super-utilisateur les ait changè. Recharger les tables de droits affecte les nouvelles connexions, mais affecte aussi les anciennes comme indiquè dans Section 4.3.3 [Privilege changes], page 232.
- Si vous n'arrivez pas á faire fonctionner votre mot de passe, souvenez-vous que vous devez utiliser la fonction `PASSWORD()` si vous le changez avec les commandes `INSERT`, `UPDATE`, ou `SET PASSWORD`. L'utilisation de la fonction `PASSWORD()` n'est pas nècessaire si vous spècifiez le mot de passe en utilisant la commande `GRANT ... INDENTIFIED BY` ou la commande `mysqladmin password`. Voir Section 4.3.7 [Passwords], page 237.
- `localhost` est un synonyme de votre nom d'hôte local, et est aussi l'hôte par dèfaut auquel le client essaye de se connecter si vous n'en spècifiez pas un explicitement. Toutefois, les connexions á `localhost` ne fonctionnent pas si vous utilisez une version antèrieure á la 3.23.27 qui utilise les MIT-pthreads (les connexions á `localhost` sont rèalisèes en utilisant les sockets Unix, qui n'ètaient pas supportès par les MIT-pthreads á ce moment). Pour contourner ce probléme sur de tels systémes, vous devez utiliser l'option `--host` pour nommer l'hôte du serveur explicitement. Cela crèera une connexion TCP/IP vers le serveur `mysqld`. Dans ce cas, vous devez avoir votre vrai nom d'hôte dans les entrèes de la table `user` du serveur hôte. (Cela est vrai même si vous utilisez un programme client sur la même machine que le serveur.)
- Si vous obtenez une erreur `Access denied` lorsque vous essayez de vous connecter á la base de donnèes avec `mysql -u nom_utilisateur nom_base`, vous pouvez avoir un probléme dans la table `user`. Vèrifiez le en vous exècutant `mysql -u root mysql` et entrant la commande SQL suivante :

  ```
  mysql> SELECT * FROM user;
  ```

  Le rèsultat devrait comprendre une entrèe avec les colonnes `Host` et `User` correspondate au nom d'hôte de votre ordinateur et votre nom d'utilisateur MySQL.
- Le message d'erreur `Access denied` vous dira en tant que qui vous essayez de vous identifier, l'hôte á partir duquel vous voulez le faire, et si vous utilisez ou pas un mot de passe. Normalement, vous devez avoir une entrèe dans la table `user` qui correspondent au nom d'hôte et nom d'utilisateur donnès dans le message d'erreur. Par exemple, si vous obtenez une erreur qui contient `Using password: NO`, cela signifie que vous avez essayè de vous connecter sans mot de passe.
- Si vous obtenez l'erreur suivante en essayant de vous connecter á partir d'un hôte diffèrenet de celui sur lequel est placè le serveur, c'est qu'il n'y a pas d'enregistrement dans la table `user` qui correspond á cet hôte :

  ```
  Host ... is not allowed to connect to this MySQL server
  ```

  Vous pouvez contourner le probléme en utilisant l'outil en ligne de commande `mysql` (sur l'hôte du serveur !) pour ajouter un enregistrement aux tables `user`, `db`, ou `host` pour la combinaison utilisateur / nom d'hôte avec laquelle vous essayez de vous connecter puis exècuter `mysqladmin flush-privileges`. Si vous n'avez pas la version 3.22 de MySQL et que vous ne connaissez ni l'IP ni le nom d'hôte á partir duquel vous essayez de vous connecter, vous devez crèer une entrèe avec `'%'` dans la colonne `Host` dans la table `user` et redèmarrer `mysqld` avec l'option `--log` sur la machine serveur.

Aprés avoir essayè á nouveau de vous connecter á partir de la machine cliente, les informations contenues dans le log de MySQL vous apprendront comment vous vous êtes vraiment connectès. (Remplacez alors l'entrèe de la table `user` contenant `'%'` avec le nom d'hôte qui apparait dans le log. Sinon, vous aurez un systéme non-sècurisè.)

Une autre raison pour cette erreur sous Linux est que vous utilisez une version binaire de MySQL qui est compilèe avec une version de glibc diffèrente de celle que vous utilisez. Dans ce cas, vous devez soit mettre á jour votre systéme d'exploitation/glibc, soit tèlècharger les sources de MySQL et les compiler vous-même. Un RPM de sources est normalement facile á compiler et installer, cela ne devrait donc pas vous poser de gros problémes.

- Si vous obtenez une erreur oú le nom d'hôte est absent ou que celui-ci est une adresse IP alors que vous avez bien entrè le nom d'hôte :

```
shell> mysqladmin -u root -pxxxx -h some-hostname ver
Access denied for user: 'root@' (Using password: YES)
```

  Cela signifie que MySQL a rencontrè des erreurs lors de la rèsolution de l'IP du nom d'hôte. Dans ce cas, vous pouvez exècuter `mysqladmin flush-hosts` pour vider le cache interne des DNS. Voir Section 5.5.5 [DNS], page 395.

  Les autres solutions sont :

  – Essayez de trouver le probléme avec votre serveur DNS et corrigez le.
  – Spècifiez les IP á la place des noms d'hôtes dans les tables de droits de MySQL.
  – Dèmarrez `mysqld` avec `--skip-name-resolve`.
  – Dèmarrez `mysqld` avec `--skip-host-cache`.
  – Connectez vous á `localhost` si vous utilisez le serveur et le client sur la même machine.
  – Placez le nom de la machine cliente dans `/etc/hosts`.

- Si `mysql -u root test` fonctionne mais que `mysql -h votre_hote -u root test` provoque une erreur `Access denied`, il se peut que vous ayez entrè de mauvaises informations pour votre nom d'hôte dans la table `user`. Un probléme commun ici est que la valeur `Host` dans la table `user` spècifie un nom d'hôte non-qualifiè, mais que vos routines systéme de rèsolution de noms retournent un nom de domaine pleinement qualifiè (ou vice-versa). Par exemple, si vous avez une entrèe avec l'hôte `'tcx'` dans la table `user`, mais que vos DNS disent á MySQL que votre nom d'hôte est `'tcx.subnet.se'`, l'entrèe ne fonctionnera pas. Essayez d'ajouter une entrèe dans la table `user` qui contient votre adresse IP en tant que valeur de la colonne `Host`. (Une alternative est d'ajouter une entrèe dans la table `user` avec une valeur de `Host` qui contient un caractére spècial, par exemple, `'tcx.%'`. Toutefois, l'utilisation des noms d'hôtes se terminant par '`%`' est **non-sècurisè** et **n**'est **pas** recommendè !)
- Si `mysql -u utilisateur test` fonctionne mais que `mysql -u utilisateur autre_base` ne fonctionne pas, vous n'avez pas d'entrèe pour `autre_base` listèe dans la table `db`.
- Si `mysql -u utilisateur nom_base` fonctionne á partir du serveur, mais que `mysql -h nom_hote -u utilisateur nom_base` ne fonctionne pas á partir d'une autre machine, cette machine n'est pas listèe dans la table `user` ou `db`.

- Si vous n'arrivez pas á trouver pourquoi vous obtenez l'erreur `Access denied`, effacez toutes les entrèes de la table `user` dont la valeur du champ `Host` contiennent des caractéres spèciaux (entrèes contenant '`%`' ou '`_`'). Une erreur commune est d'insèrer une nouvelle entrèe avec `Host='%'` et `User='un utilisateur'`, en pensant que cela vous permettra de spècifier `localhost` pour vous connecter á partir de la même machine. La raison pour laquelle cela ne fonctionnera pas est que les droits par dèfaut incluent une entrèe avec `Host='localhost'` et `User=''`. Puisque cette entrèe posséde une valeur de `Host` ègale á `'localhost'`, qui est plus spècifique que `'%'`, elle est utilisèe de prèfèrence á la nouvelle entrèe lors de la connexion á partir de `localhost` ! La procèdure correcte est d'insèrer une seconde entrèe avec `Host='localhost'` et `User='un_utilisateur'`, ou de supprimer l'entrèe avec `Host='localhost'` et `User=''`.
- Si vous avez l'erreur suivante, vous avez peut-être un probléme avec la table `db` ou `host` :

  ```
  Access to database denied
  ```

  Si l'entrèe sèlectionnèe dans la table `db` posséde un champ `Host` vide, assurez-vous qu'il y'a au moins une entrèe correspondante dans la table `host` spècifiant les hôtes auxquels l'entrèe dans la table `db` s'applique.

  Si vous obtenez l'erreur lors de l'utilisation des commandes SQL `SELECT ... INTO OUTFILE` ou `LOAD DATA INFILE`, votre entrèe dans la table `user` ne posséde probablement pas le privilége `FILE`.
- Souvenez-vous que le programme client utilise les paramétres de connexion spècifiès dans les fichiers de configuration ou des les variables d'environnement. Voir Annexe F [Environment variables], page 828. Si un client semble envoyer les mauvais paramétres de connexion par dèfaut lorsque vous ne les spècifiez pas en ligne de commande, vèrifiez votre environnement et le fichier '`.my.cnf`' dans votre rèpertoire personnel. Vous devrez aussi vèrifier les fichiers systéme de configuration de MySQL, bien qu'il est fort peu probable que les paramétres de connexion du client y soient spècifiès. Voir Section 4.1.2 [Option files], page 198. Si vous obtenez une erreur `Access denied` quand vous dèmarrer un client sans aucune option, assurez-vous de ne pas avoir spècifiè un vieux mot de passe dans l'un de vos fichiers d'options ! Voir Section 4.1.2 [Option files], page 198.
- Si vous apportez des modifications au tables de droits directement (en utilisant une requête `INSERT` ou `UPDATE`) et que vos changements semblent ignorès, souvenez vous que vous devez exècuter une requête `FLUSH PRIVILEGES` ou la commande `mysqladmin flush-privileges` pour demander au serveur de lire á nouveau les tables de droits. Sinon, vos changements ne seront pris en compte qu'au prochain dèmarrage du serveur. Souvenez-vous qu'aprés avoir choisi le mot de passe `root` avec une commande `UPDATE`, vous n'aurez pas á le spècifier avant de recharger les priviléges, car le serveur ne sait pas que vous l'avez modifiè !
- Si vous avez des problémes d'accés avec un programme Perl, PHP, Python, ou ODBC, essayez de vous connecter au serveur avec `mysql -u utilisateur nom_base` ou `mysql -u utilisateur -pvotre_passe nom_base`. Si vous pouvez vous connecter en utilisant le client `mysql`, c'est que le probléme vient de votre programme et non des droits MySQL. (Notez qu'il n'y a pas d'espace entre `-p` et le mot de passe; vous pouvez aussi

utiliser la syntaxe `--password=votre_passe` pour spècifier le mot de passe. Si vous utilisez l'option `-p` toute seule, MySQL vous demandera le mot de passe.)

- Pour les tests, dèmarrez le dèmon `mysqld` avec l'option `--skip-grant-tables`. Vous pourrez alors changer les tables de droits MySQL puis utiliser le script `mysqlaccess` pour vèrifier si vos changements ont l'effet dèsirè. Lorsque vous êtes satisfait de vos modifications, exècutez `mysqladmin flush-privileges` pour dire au serveur `mysqld` de commencer á utiliser les nouvelles tables de droits. **Note** : Recharger les tables de droits ècrase l'option `--skip-grant-tables`. Cela vous permet de dire au serveur de commencer á prendre en considèration les droits sans avoir á le couper et le redèmarrer.
- Si rien ne fonctionne, dèmarrez le dèmon `mysqld` avec l'option de dèbogage (par exemple, `--debug=d,general,query`). Cela affichera l'hôte et les informations de l'utilisateur pour chaque tentative de connexion. Les informations á propos de chaque commande exècutèe seront aussi affichèes. Voir Section E.1.2 [Making trace files], page 816.
- Si vous avez d'autres problémes avec les tables de droits de MySQL et que vous sentez que vous devez envoyer le probléme á la liste de diffusion, fournissez toujours le contenu de vos tables de droits. Vous pouvez obtenir les donnèes avec la commande `mysqldump mysql`. Comme toujours, postez votre probléme á l'aide du script `mysqlbug`. Voir Section 1.6.2.3 [Bug reports], page 27. Dans certains cas, vous aurez besoin de redèmarrer `mysqld` avec `--skip-grant-tables` pour pouvoir exècuter `mysqldump`.

4.3 Gestion des comptes utilisateurs de MySQL

4.3.1 Syntaxe de `GRANT` et `REVOKE`

```
GRANT priv_type [(liste_colonnes)] [, priv_type [(liste_colonnes)] ...]
    ON {nom_de_table | * | *.* | nom_base.*}
    TO nom_utilisateur [IDENTIFIED BY [PASSWORD] 'password']
        [, nom_utilisateur [IDENTIFIED BY 'password'] ...]
    [REQUIRE
        NONE |
     [{SSL| X509}]
[CIPHER cipher [AND]]
[ISSUER issuer [AND]]
[SUBJECT subject]]
    [WITH [GRANT OPTION | MAX_QUERIES_PER_HOUR # |
                          MAX_UPDATES_PER_HOUR # |
                          MAX_CONNECTIONS_PER_HOUR #]]

REVOKE priv_type [(liste_colonnes)] [, priv_type [(liste_colonnes)] ...]
    ON {nom_de_table | * | *.* | nom_base.*}
    FROM nom_utilisateur [, nom_utilisateur ...]
```

`GRANT` est disponible depuis MySQL version 3.22.11. Pour les versions plus anciennes de MySQL, la commande `GRANT` ne fait rien.

Les commandes `GRANT` et `REVOKE` permettent aux administrateurs systéme de crèer des utilisateurs, de leur donner ou de leur retirer des droits, sur 4 niveaux :

Niveau global
: Les droits globaux s'appliquent á toutes les bases de donnèes d'un serveur. Ces droits sont stockès dans la table `mysql.user`.

Niveau base de donnèes
: Les droits de niveau de base de donnèes s'appliquent á toutes les tables d'une base de donnèes. Ces droits sont stockès dans les tables `mysql.db` et `mysql.host`.

Niveau table
: Les droits de table s'appliquent á toutes les colonnes d'une table. Ces droits sont stockès dans la table `mysql.tables_priv`.

Niveua de colonne
: Les droits de niveau de colonnes s'appliquent á des colonnes dans une table. Ces droits sont stockès dans la table `mysql.columns_priv`.

Si vous donnez des droits á un utilisateur qui n'existe pas, vous crèerez cet utilisateur. Pour voir des illustrations du fonctionnement de la commande `GRANT`, voyez Section 4.3.5 [Adding users], page 233.

Pour les commandes `GRANT` et `REVOKE`, la clause `priv_type` peut être spècifiée par les constantes suivantes :

`ALL [PRIVILEGES]`	Tous les droits sauf `WITH GRANT OPTION`.
`ALTER`	Autorise l'utilisation de `ALTER TABLE`.
`CREATE`	Autorise l'utilisation de `CREATE TABLE`.
`CREATE TEMPORARY TABLES`	Autorise l'utilisation de `CREATE TEMPORARY TABLE`.
`DELETE`	Autorise l'utilisation de `DELETE`.
`DROP`	Autorise l'utilisation de `DROP TABLE`.
`EXECUTE`	Autorise l'utilisateur á exècuter des procèdures stockèes (pour MySQL 5.0).
`FILE`	Autorise l'utilisation de `SELECT ... INTO OUTFILE` et `LOAD DATA INFILE`.
`INDEX`	Autorise l'utilisation de `CREATE INDEX` et `DROP INDEX`.
`INSERT`	Autorise l'utilisation de `INSERT`.
`LOCK TABLES`	Autorise l'utilisation de `LOCK TABLES` sur les tables pour lesquelles l'utilisateur a les droits de `SELECT`.
`PROCESS`	Autorise l'utilisation de `SHOW FULL PROCESSLIST`.
`REFERENCES`	Rèservè pour le futur.
`RELOAD`	Autorise l'utilisation de `FLUSH`.
`REPLICATION CLIENT`	Donne le droit á l'utilisateur de savoir oú sont les maîtres et esclaves.
`REPLICATION SLAVE`	Nècessaire pour les esclaves de rèplication (pour lire les historiques binaires du maître).
`SELECT`	Autorise l'utilisation de `SELECT`.
`SHOW DATABASES`	`SHOW DATABASES` affiche toutes les bases de donnèes.
`SHUTDOWN`	Autorise l'utilisation de `mysqladmin shutdown`.

SUPER	Autorise une connexion unique même si max_connections est atteind, et l'exècution des commandes CHANGE MASTER, KILL thread, mysqladmin debug, PURGE MASTER LOGS et SET GLOBAL.
UPDATE	Autorise l'utilisation de UPDATE.
USAGE	Synonyme de "pas de droits".

USAGE peut être utilisè lorsque vous voulez crèer un utilisateur sans aucun droit.

Les droits de CREATE TEMPORARY TABLES, EXECUTE, LOCK TABLES, REPLICATION ..., SHOW DATABASES et SUPER sont nouveaux en version 4.0.2. Pour utiliser ces droits aprés mise á jour en 4.0.2, vous devez exècuter le script mysql_fix_privilege_tables.

Dans les anciennes versions de MySQL, le droit de PROCESS donnait les mêmes droits que le nouveau droit SUPER.

Pour retirer le droit de GRANT á un utilisateur, utilisez les mêmes valeurs de priv_type avec GRANT OPTION :

```
mysql> REVOKE GRANT OPTION ON ... FROM ...;
```

Les valeurs de priv_type que vous pouvez spècifier pour une table sont SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, GRANT, INDEX et ALTER.

Les seules valeurs de priv_type que vous pouvez spècifier pour une colonne, lorsque vous utilisez la clause liste_colonnes), sont SELECT, INSERT et UPDATE.

Vous pouvez donner des droits globaux en utilisant la syntaxe ON *.*. Vous pouvez donner des droits de base en utilisant la syntaxe ON nom_base.*. Si vous spècifiez ON * et que vous avez une base de donnèes qui est dèjá sèlectionnèe, vous allez donner des droits pour la base de donnèes courante. **Attention** : si vous spècifiez ON * et que vous **n'avez pas** de base courante, vous allez affecter les droits au niveau du serveur !

Notez bien : les caractéres joker '_' and '%' sont autorisès lors de la spècification de noms dans la commande GRANT. Cela signifie que si vous voulez utiliser par exemple le caractére littèral '_' comme nom de base, vous devez le spècifier sous la forme '_' dans la commande GRANT, pour èviter á l'utilisateur d'accèder á d'autres bases, dont le nom pourrait correspondre au masque d'expression règuliére ainsi crèè. Utilisez plutôt GRANT ... ON 'foo_bar'.* TO ....

Afin de permettre l'identification des utilisateurs depuis des hôtes arbitraires, MySQL supporte la spècification du nom d'utilisateur nom_utilisateur sous la forme user@host. Si vous voulez spècifier un nom d'utilisateur user qui contient des caractéres spèciaux tels que '-', ou une chaîne d'hôte host qui contient des caractéres joker (comme '%'), vous pouvez placer le nom de l'utilisateur ou de l'hôte entre guillemets (par exemple, 'test-utilisateur'@'test-nomdhote').

Vous pouvez spècifier des caractéres jokers dans le nom d'hôte. Par exemple, user@'%.loc.gov' fait correspondre l'utilisateur user de n'importe quel hôte du domaine loc.gov, et user@'144.155.166.%' fait correspondre l'utilisateur user á n'importe quelle adresse de la classe C 144.155.166.

La forme simple de user est synonyme de user@"%".

MySQL ne supporte pas de caractéres joker dans les noms d'utilisateur. Les utilisateurs anonymes sont dèfinis par l'insertion de ligne avec User='' dans la table mysql.user, ou en crèant un utilisateur avec un nom vide, grâce á la commande GRANT.

Note : si vous autorisez des utilisateurs anonymes á se connecter á votre serveur, vous devriez aussi donner ces droits á tous les utilisateurs locaux `user@localhost` car sinon, la ligne dans la table `mysql.user` sera utilisèe lorsque l'utilisateur se connectera au serveur MySQL depuis la machine locale !

Vous pouvez vèrifier si cela s'applique á vous en exècutant la requête suivante :

```
mysql> SELECT Host,User FROM mysql.user WHERE User='';
```

Actuellement, la commande `GRANT` supporte uniquement les noms d'hôte, colonne, table et bases de donnèes d'au plus 60 caractéres. Un nom d'utilisateur peut être d'au plus 16 caractéres.

Les droits pour les tables et colonnes sont combinès par OU logique, avec les quatre niveaux de droits. Par exemple, si la table `mysql.user` spècifie qu'un utilisateur a un droit global de `SELECT`, ce droit ne pourra pas être annulè au niveau base, table ou colonne.

Les droits d'une colonne sont calculès comme ceci :

```
droit global
OR (droit de base de donnèes ET droit d'hôte)
OR droit de table
OR droit de colonne
```

Dans la plupart des cas, vous donnez des droits á un utilisateur en utilisant un seul des niveaux de droits ci-dessus, ce qui fait que la vie n'est pas aussi compliquèe. Le dètails de la procèdure de vèrification des droits et disponible dans Section 4.2 [Privilege system], page 203.

Si vous donnez des droits á une paire utilisateur/hôte qui n'existe pas dans la table `mysql.user`, une ligne sera crèèe et restera disponible jusqu'á son effacement avec la commande `DELETE`. En d'autre termes, `GRANT` crèe une ligne dans la table `user`, mais `REVOKE` ne la supprime pas. Vous devez le faire explicitement avec la commande `DELETE`.

Avec MySQL version 3.22.12 ou plus rècent, si un nouvel utilisateur est crèè, ou si vous avez les droits de GRANT globaux, le mot de passe sera configurè avec le mot de passe spècifiè avec la clause `IDENTIFIED BY`, si elle est fournie. Si l'utilisateur a dèjá un mot de passe, il sera remplacè par ce nouveau.

Si vous ne voulez pas faire passer le mot de passe en texte clair, vous pouvez immèdiatement utiliser l'option `PASSWORD` suivi du mot de passe dèjá chiffrè avec la fonction `PASSWORD()` ou l'API C `make_scrambled_password(char *to, const char *password)`.

Attention : si vous crèez un nouvel utilisateur, mais que vous ne spècifiez pas la clause `IDENTIFIED BY`, l'utilisateur n'aura pas de mot de passe. Ce n'est pas sûr.

Les mots de passe peuvent aussi être modifiès par la commande `SET PASSWORD`. Voir Section 5.5.6 [SET], page 396.

Si vous donnez les droits de base, une ligne sera ajoutèe dans la table `mysql.db`. Lorsque les droits sur cette base seront supprimès avec la commande `REVOKE`, cette ligne disparaîtra.

Si un utilisateur n'a pas de droits sur une table, elle ne sera pas affichèe lorsqu'il requiert la liste des tables avec la commande `SHOW TABLES`.

La clause `WITH GRANT OPTION` donne á l'utilisateur le droit de donner les droits qu'il posséde á d'autres utilisateurs. La plus grande prudence est recommandèe pour cette commande, car il permettra á terme á deux utilisateurs de combiner les droits dont ils disposent.

`MAX_QUERIES_PER_HOUR #`, `MAX_UPDATES_PER_HOUR #` et `MAX_CONNECTIONS_PER_HOUR #` sont nouveaux en MySQL 4.0.2. Ces deux options limitent le nombre de requêtes et de modifications qu'un utilisateur peut rèclamer dans une heure. Si `#` vaut 0 (valeur par dèfaut), alors cela signifie qu'il n'y a pas de limitations pour cet utilisateur. Voir Section 4.3.6 [User resources], page 236.

Vous ne pouvez pas donner á un autre utilisateur un droit que vous ne possèdez pas vous-même. Le droit de `GRANT` vous donne le droit de diffuser les droits dont vous disposez dèjá.

Soyez bien conscient que lorsque vous donnerez le droit de `GRANT` á un niveau particulier, tous les droits qu'un utilisateur posséde dèjá (oú qui lui seront donnès dans le futur) seront alors diffusables á d'autres individus. Supposons que vous donnez les droits de `INSERT` á un utilisateur, pour une table. Si vous ajoutez le droit de `WITH GRANT OPTION`, cet utilisateur peut donner le droit de `SELECT` mais aussi celui d'`INSERT`. Si vous donnez aussi le droit de `UPDATE`, il pourra alors diffuser les droits de `INSERT`, `SELECT` et `UPDATE`.

Il est recommandè de ne pas donner les droits de `ALTER` á un utilisateur normal. Si vous le faites, les utilisateurs pourront alors contourner le systéme de droits en renommant des tables !

Notez que si vous utilisez des droits de niveau table ou colonne même pour un utilisateur, le serveur vèrifiera alors ces droits pour tous les utilisateurs, et cela ralentira MySQL un peu.

Lorsque `mysqld` dèmarre, tous les droits sont stockès en mèmoire. Les droits de bases, tables et colonnes prennent aussitôt effet, et les droits des utilisateurs prendront effet dés leur prochaine configuration. Les modifications sur les tables de droits que vous effectuez avec les commandes `GRANT` et `REVOKE` sont prises en compte immèdiatement par le serveur. Si vous modifiez manuellement les tables (avec `INSERT`, `UPDATE`, etc...), vous devez exècuter la commande `FLUSH PRIVILEGES`, ou la commande en ligne `mysqladmin flush-privileges` pour indiquer au serveur qu'il doit recharger les droits. Voir Section 4.3.3 [Privilege changes], page 232.

Les diffèrences notables entre l'ANSI SQL et MySQL pour la commande `GRANT` sont :

- Les droits MySQL sont donnès pour une combinaison nom d'utilisateur + nom d'hôte, et non pas pour un nom d'hôte seulement.
- L'ANSI SQL n'a pas de droits globaux ou de niveau base de donnèes, et l'ANSI SQL ne supporte pas tous les types de droits que MySQL supporte. MySQL ne supporte pas le droit ANSI SQL de `TRIGGER` ou `UNDER`.
- Les droits ANSI SQL sont structurès de maniére hièrarchique. Si vous supprimez un utilisateur, tous les droits donnès á cet utilisateur seront supprimès. Avec MySQL, les droits ne sont pas automatiquement supprimès, et vous devez les supprimer manuellement, si besoin.
- Avec MySQL, si vous avez le droit de `INSERT` sur uniquement quelques colonnes de la table, vous pourrez exècuter des insertions. Les colonnes pour lesquelles vous n'avez pas de droit prendront alors leur valeur par dèfaut. L'ANSI SQL vous impose d'avoir les droits d'`INSERT` sur toutes les colonnes.
- Lorsque vous dètruisez une table avec ANSI SQL, tous les droits liès á la table sont supprimès. Si vous supprimez un droit en ANSI SQL, tous les droits qui ètaient basès sur ce

droit sont supprimès. Avec MySQL, les droits peuvent être abandonnès explicitement avec la commande `REVOKE`, ou en manipulant les tables de droits de MySQL.

Pour une description de l'utilisation de `REQUIRE`, voyez Section 4.3.9 [Secure connections], page 239.

4.3.2 Nom d'utilisateurs MySQL et mots de passe

Il y a de nombreuses diffèrences entre les utilisations des noms et mots de passe sous MySQL, et celles qui sont faites sous Unix ou Windows :

- Les noms d'utilisateurs, tels qu'utilisès pour le processus d'identification sous MySQL, n'ont rien á voir avec les noms d'utilisateurs Unix ou Windows. La plupart des clients utilisent par dèfaut leur mot de passe Unix, mais c'est surtout parce que c'est pratique. Les programmes clients permettent d'utiliser des noms d'utilisateurs diffèrents avec les options `-u` et `--user`. Cela signifie que vous ne pouvez pas rendre une base de donnèes sècuritaire sans donner de mots de passe á tous les clients. Tout le monde peut essayer de se connecter au serveur sous n'importe quel nom, et il sera possible de se connecter si un nom d'utilisateur n'a pas de mot de passe.
- Les noms d'utilisateurs MySQL peuvent avoir jusqu'á 16 caractéres ; les noms d'utilisateurs Unix sont gènèralement limitès á 8 caractéres.
- Les mots de passe MySQL n'ont aucun rapport avec le passeport Unix. Il n'y a pas nècessairement de connexion entre le mot de passe que vous utilisez pour vous connecter sur la machine Unix et celui que vous utilisez pour accèder au serveur MySQL.
- MySQL chiffre les mots de passe avec un algorithme diffèrent de celui qui est utilisè par Unix. Reportez-vous aux descriptions des fonctions `PASSWORD()` et `ENCRYPT()` dans Section 6.3.6.2 [Miscellaneous functions], page 472. Notez que même si le mot de passe est enregistrè 'brouillè', connaître votre mot de passe 'brouillè' est suffisant pour se connecter au serveur MySQL.

Les utilisateurs MySQL et leurs droits sont gènèralement crèès par la commande `GRANT`. Voir Section 4.3.1 [GRANT], page 226.

Lorsque vous vous connectez á un serveur MySQL avec un client en ligne de commande, vous devez spècifier le mot de passe avec l'option `--password=mot-de-passe`. Voir Section 4.2.8 [Connecting], page 215.

```
mysql --user=monty --password=devine nom_base
```

Si vous voulez que le client vous demande le mot de passe á la volèe, vous devriez utiliser l'option `--password` sans argument :

```
mysql --user=monty --password nom_base
```

ou la version courte :

```
mysql -u monty -p nom_base
```

Notez que dans le dernier exemple, le mot de passe n'est **pas** 'nom_base'.

Si vous voulez utiliser l'option `-p` pour fournir un mot de passe, vous pouvez aussi essayer ceci :

```
mysql -u monty -pdevine nom_base
```

Sur certains systémes, l'appel systéme que MySQL utiliser pour demander un mot de passe va automatiquement couper le mot de passe á 8 caractéres. En interne, MySQL n'a pas de limite á la taille du mot de passe.

4.3.3 Quand les modifications de priviléges prennent-ils effets ?

Lorsque `mysqld` est lancè, toutes les tables de droits sont lues, et sont utilisèes.

Les modifications aux tables de droits que vous faites avec `GRANT`, `REVOKE` et `SET PASSWORD` sont immèdiatement prises en compte par le serveur.

Si vous modifiez les tables de droits manuellement (avec `INSERT`, `UPDATE`, etc...), vous devez exècuter la commande `FLUSH PRIVILEGES` ou la commande `mysqladmin flush-privileges`, ou encore `mysqladmin reload` pour dire au serveur de relire les tables de droits. Sinon, vos modifications n'auront *aucun effet* jusqu'au redèmarrge du serveur. Si vous modifiez les tables de droits manuellement, mais que vous oubliez de recharger les droits, vous vous demanderez sûrement pourquoi vos modifications n'ont pas d'effet.

Lorsque le serveur remarque que les tables de droits ont ètè modifièes, les connexions existantes avec les clients sont modifièes comme ceci :

- Les droits de table et colonnes prennent effet á la prochaine requête du client.
- Les droits de bases prennent effet á la prochaine commande `USE nom_de_base`.
- Les droits globaux et les modifications de droits prennent effets lors de la prochaine connexion.

4.3.4 Crèation des premiers droits MySQL

Aprés avoir installè MySQL, vous devez configurer les premiers droits du serveur en exècutant le script `scripts/mysql_install_db`. Voir Section 2.3.1 [Quick install], page 85. Le script `mysql_install_db` dèmarre le serveur `mysqld` et initialise les tables de droits, avec les paramétres suivants :

- L'utilisateur MySQL `root` est crèè en tant qu'administrateur avec tous les droits. Les connexions doivent se faire depuis l'hôte local.

  Note : Le mot de passe de l'utilisateur initial `root` est vide, ce qui permet á n'importe qui de se connecter en tant que `root` *sans mot de passe*, pour profiter de tous les droits.
- Un utilisateur anonyme est crèè, qui peut faire ce qu'il veut avec toutes les tables dans la base de donnèes `'test'` ou commençant par `'test_'`. Les connexions doivent être faites depuis l'hôte local. Cela signifie qu'un utilisateur local peu se connecter sans mot de passe et être traitè comme un utilisateur anonyme.
- Les autres droits sont rèservès. Par exemple, un utilisateur normal ne peut pas utiliser les commandes `mysqladmin shutdown` ou `mysqladmin processlist`.

Note : les droits par dèfaut sont diffèrents sous Unix. Voir Section 2.6.2.3 [Windows running], page 123.

Comme votre installation de base n'est pas du tout sècurisèe, la premiére action que vous devez prendre est de spècifier un mot de passe á votre utilisateur `root`. Vous pouvez faire cela comme ceci (notez que vous aurez á spècifier le mot de passe en utilisant la fonction `PASSWORD()`) :

```
shell> mysql -u root mysql
mysql> SET PASSWORD FOR root@localhost=PASSWORD('new_password');
```

Si vous savez ce que vous faites, vous pouvez aussi directement manipuler les tables de droits :

```
shell> mysql -u root mysql
mysql> UPDATE user SET Password=PASSWORD('new_password')
    ->     WHERE user='root';
mysql> FLUSH PRIVILEGES;
```

Une autre mèthode pour modifier de mot de passe est d'utiliser la commande en ligne `mysqladmin` :

```
shell> mysqladmin -u root password new_password
```

Seuls les utilisateurs ayant des droits d'accés en lecture et ècriture dans la base de donnèes `mysql` peuvent modifier les mots de passe des autres utilisateurs. Tous les utilisateurs normaux, á l'exception des anonymes, peuvent uniquement modifier leur mot de passe personnel, avec l'une des commandes ci-dessus ou avec la commande `SET PASSWORD=PASSWORD('new password')`.

Notez que si vous modifiez le mot de passe dans la table `user` en utilisant la premiére mèthode, vous devez indiquer au serveur qu'il doit relire les droits avec la commande `FLUSH PRIVILEGES`), sinon ces modifications ne seront pas prises en compte.

Une fois que le mot de passe `root` a ètè modifiè, vous devrez le fournir pour vous connecter au serveur en tant que `root`.

Vous pouvez aussi laisser le mot de passe du `root` vide, afin de faire des connexions facilement lors de la configuration initiale du serveur. Mais n'oubliez pas de le spècifier lors de la mise en production.

Voyez le script `scripts/mysql_install_db` pour savoir comment il configure les droits par dèfaut. Vous pouvez l'utiliser comme base pour ajouter des droits á d'autres utilisateurs.

Si vous voulez que les droits initiaux soient diffèrents, vous pouvez aussi modifier le script `mysql_install_db` avant de l'exècuter.

Pour recrèer complétement les tables de droits, effacez tous les fichiers '`.frm`', '`.MYI`' et '`.MYD`' dans le dossier contenant les tables de la base `mysql`. (C'est le dossier '`mysql`' dans le dossier de donnèes, qui est indiquè par la commande `mysqld --help`.) Puis, exècutez le script `mysql_install_db`, aprés l'avoir èventuellement modifiè avec les droits que vous souhaitez.

Note : pour les versions de MySQL antèrieures á la version 3.22.10, il ne faut pas effacer les fichiers '`.frm`'. Si vous le fait accidentellement, vous devez les copier depuis la distribution MySQL avant d'exècuter le script `mysql_install_db`.

4.3.5 Ajouter de nouveaux utilisateurs á MySQL

Vous pouvez ajouter des utilisateurs de deux façons diffèrentes : en utilisant la commande `GRANT` ou en manipulant la table des droits de MySQL directement. La mèthode prèfèrèe consiste á utiliser la commande `GRANT`, car elle est plus concise et qu'il y'a moins de risques d'erreurs. Voir Section 4.3.1 [GRANT], page 226.

Il y'a aussi beaucoup de programmes contribuès comme `phpmyadmin` qui peuvent être utilisès pour crèer et administrer les utilisateurs. Voir Section 1.6.1 [Portals], page 23.

Les exemples suivants montrent comment utiliser le client `mysql` pour crèer de nouveaux utilisateurs. Ces exemples supposent que les priviléges sont attribuès en accord avec les valeurs par dèfaut discutèes dans la section prècèdente. Cela signifie que pour effectuer des changements, vous devez être sur la même machine oú `mysqld` tourne, vous devez vous connecter en tant qu'utilisateur MySQL `root`, et l'utilisateur `root` doit avoir le droit `INSERT` sur la base `mysql` et le droit d'administration `RELOAD`. Si vous avez changè le mot de passe de l'utilisateur `root`, vous devez le spècifier dans les commandes `mysql` ci-dessous.

Vous pouvez ajouter de nouveaux utilisateurs en utilisant des commandes `GRANT` :

```
shell> mysql --user=root mysql
mysql> GRANT ALL PRIVILEGES ON *.* TO monty@localhost
    ->     IDENTIFIED BY 'un_mot_de_passe' WITH GRANT OPTION;
mysql> GRANT ALL PRIVILEGES ON *.* TO monty@"%"
    ->     IDENTIFIED BY 'un_mot_de_passe' WITH GRANT OPTION;
mysql> GRANT RELOAD,PROCESS ON *.* TO admin@localhost;
mysql> GRANT USAGE ON *.* TO dummy@localhost;
```

Ces commandes `GRANT` ajoutent trois nouveaux utilisateurs :

`monty` Un super-utilisateur qui peut se connecter au serveur d'oú il veut, mais qui doit utiliser le mot de passe `'un_mot_de_passe'` pour le faire. Notez que nous devons exècuter une commande `GRANT` pour `monty@localhost` et `monty@"%"`. Si nous n'ajoutons pas l'entrèe avec `localhost`, l'entrèe concernant l'utilisateur anonyme pour `localhost` qui est crèèe par `mysql_install_db` prendra prècèdence lors de la connexion á partir de l'hôte local, car elle a une entrèe plus spècifique pour la valeur du champ `Host` et de plus, elle vient en premier dans l'ordre de tri de la table `user`.

`admin` Un utilisateur qui peut se connecter depuis `localhost` sans mot de passe et qui a les droits administratifs `RELOAD` et `PROCESS`. Cela permet á cet utilisateur d'exècuter les commandes `mysqladmin reload`, `mysqladmin refresh`, et `mysqladmin flush-*`, ainsi que `mysqladmin processlist`. Aucun droit liè aux bases de donnèes n'est donnè. (Ils peuvent l'êetre plus tard en utilisant d'autres instructions `GRANT`.)

`dummy` Un utilisateur qui peut se connecter sans mot de passe, mais seulement á partir de l'hôte local. Les droits globaux sont tous á `'N'`—le type de droit `USAGE` vous permet de crèer un utilisateur dèmuni de priviléges. Il est supposè que vous lui assignerez les droits spècifiques aux bases de donnèes plus tard.

Vous pouvez ajouter les mêmes droits d'accés aux utilisateurs en utilisant directement des requêtes `INSERT` puis en demandant au serveur de recharger les tables de droits :

```
shell> mysql --user=root mysql
mysql> INSERT INTO user VALUES('localhost','monty',PASSWORD('un_mot_de_passe'),
    ->          'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO user VALUES('%','monty',PASSWORD('un_mot_de_passe'),
    ->          'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO user SET Host='localhost',User='admin',
```

```
    ->           Reload_priv='Y', Process_priv='Y';
mysql> INSERT INTO user (Host,User,Password)
    ->                  VALUES('localhost','dummy','');
mysql> FLUSH PRIVILEGES;
```

Selon votre version de MySQL, vous pouvez avoir un nombre diffèrent de valeurs `'Y'` plus haut (les versions antèrieures á la 3.22.11 possédent moins de colonnes de priviléges). Pour l'utilisateur `admin`, la syntaxe d'`INSERT` ètendue la plus lisible disponible depuis la version 3.22.11 est utilisèe.

Notez que pour ajouter un super-utilisateur, vous avez juste besoin de crèer une entrèe dans la table `user` avec tous les champs de droits á `'Y'`. Aucune entrèe n'est requise dans les tables `db` et `host`.

Les colonnes de priviléges de la table `user` n'ètaient pas renseignèes explicitement dans la derniére requête `INSERT` (pour l'utilisateur `dummy`), ses colonnes prennent donc la valeur par dèfaut, `'N'`. C'est la même chose que ce que fait `GRANT USAGE`.

L'exemple suivant ajoute un utilisateur `custom` qui peut se connecter á partir des hôtes `localhost`, `server.domain`, et `whitehouse.gov`. Il ne pourra accèder á la base de donnèes `bankaccount` qu'á partir de `localhost`, á la base `expenses` qu'á partir de `whitehouse.gov`, et á la base `customer` á partir des trois hôtes. Il utilisera le mot de passe `stupid` pour les trois hôtes.

Pour configurer les priviléges de cet utilisateur en utilisant des commandes `GRANT`, exècutez ce qui suit :

```
shell> mysql --user=root mysql
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
    ->     ON bankaccount.*
    ->     TO custom@localhost
    ->     IDENTIFIED BY 'stupid';
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
    ->     ON expenses.*
    ->     TO custom@whitehouse.gov
    ->     IDENTIFIED BY 'stupid';
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
    ->     ON customer.*
    ->     TO custom@'%'
    ->     IDENTIFIED BY 'stupid';
```

La raison pour laquelle nous faisons deux insertions pour l'utilisateur 'custom' est que nous voulons lui donner accés á MySQL á partir de la machine locale avec les sockets Unix et á partir de la machine distante 'whitehouse.gov' via TCP/IP.

Pour règler les permissions d'accés en modifiant directement les tables de droits, exècutez ces commandes (notez l'appel á `FLUSH PRIVILEGES` á la fin) :

```
shell> mysql --user=root mysql
mysql> INSERT INTO user (Host,User,Password)
    -> VALUES('localhost','custom',PASSWORD('stupid'));
mysql> INSERT INTO user (Host,User,Password)
    -> VALUES('server.domain','custom',PASSWORD('stupid'));
mysql> INSERT INTO user (Host,User,Password)
```

```
    -> VALUES('whitehouse.gov','custom',PASSWORD('stupid'));
mysql> INSERT INTO db
    -> (Host,Db,User,Select_priv,Insert_priv,Update_priv,Delete_priv,
    ->  Create_priv,Drop_priv)
    -> VALUES
    -> ('localhost','bankaccount','custom','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO db
    -> (Host,Db,User,Select_priv,Insert_priv,Update_priv,Delete_priv,
    ->  Create_priv,Drop_priv)
    -> VALUES
    -> ('whitehouse.gov','expenses','custom','Y','Y','Y','Y','Y','Y');
mysql> INSERT INTO db
    -> (Host,Db,User,Select_priv,Insert_priv,Update_priv,Delete_priv,
    ->  Create_priv,Drop_priv)
    -> VALUES('%','customer','custom','Y','Y','Y','Y','Y','Y');
mysql> FLUSH PRIVILEGES;
```

Les trois premiéres requêtes `INSERT` ajoute les entrèes dans la table `user` qui permettent l'iutilisateur `custom` á se connecter á partir de plusieurs hôtes avec le mot de passe donnè, mais ne lui donnent aucun droit (tous les droits sont mis á la valeur par dèfaut qui est `'N'`). Les trois requêtes `INSERT` suivantes ajoutent des entrèes dans la table `db` qui autorisent `custom` á utiliser les bases de donnèes `bankaccount`, `expenses`, et `customer`, mais seulement s'il y accéde á partir de l'hôte spècifiè. Comme d'habitude, lorsqueles tables de droits sont modifièes directement, on doit demander au serveur des les recharger (avec `FLUSH PRIVILEGES`) pour que les changements soient pris en compte.

Si vous voulez donner un accés spècifique á un utilisateur á partir de n'importe quelle machine d'un domaine donnè, vous pouvez utiliser la commande `GRANT` qui suit :

```
mysql> GRANT ...
    ->     ON *.*
    ->     TO monutilisateur@"%.mondomaine.com"
    ->     IDENTIFIED BY 'monmotdepasse';
```

Pour faire la même chose en modifiant directement la table de droits, faites :

```
mysql> INSERT INTO user VALUES ('%.mondomaine.com', 'monutilisateur',
    ->             PASSWORD('monmotdepasse'),...);
mysql> FLUSH PRIVILEGES;
```

Vous pouvez aussi utiliser `xmysqladmin`, `mysql_webadmin`, et même `xmysql` pour insèrer, changer et mettre á jour les valeurs dans les tables de droits. Vous pouvez trouver ces utilitaires dans le dossier des contributions du site web de MySQL (`http://www.mysql.com/Downloads/Contrib/`).

4.3.6 Limiter les ressources utilisateurs

Depuis MySQL 4.0.2, il est possible de limiter certaines ressources accessibles á un utilisateur possible.

Jusqu'á prèsent, la seule mèthode possible pour limiter l'utilisation des ressources serveurs MySQL ètait de configurer la variable de dèmarrage `max_user_connections` avec une

valeur non nulle. Mais cette mèthode est strictement globale, et ne permet pas de faire des amènagements personnalisès, comme cela pourrait arriver aux fournisseurs de services internet.

Par consèquent, un systéme de gestion des ressources a ètè introduit, au niveau des utilisateurs :

- Nombre de requête par heure : Toutes les commandes qu'un utilisateur peut exècuter.
- Nombre de modifications par heure : Toute commande qui implique la modification d'une table ou d'une base.
- Nombre de connexion rèalisèes par heure : Le nombre de nouvelles connexions par heure.

Un utilisateur dans le contexte ci-dessus est une ligne dans la table `user`, qui est identifiè de maniére unique par les colonnes `user` et `host`.

Par dèfaut, les utilisateurs ne sont pas limitès dans l'utilisation des ressources ci-dessus, á moins que des limites ne leur soient imposèes. Ces limites peuvent être configurèes **uniquement** via la commande `GRANT (*.*)`, avec cette syntaxe :

```
GRANT ... WITH MAX_QUERIES_PER_HOUR N1
               MAX_UPDATES_PER_HOUR N2
               MAX_CONNECTIONS_PER_HOUR N3;
```

Il est possible de spècifier toute combinaison de clauses ci-dessus. N1, N2 et N3 sont des entiers et reprèsentent le compte par heure.

Si l'utilisateur atteint l'une des limites ci-dessus dans une heure, sa connexion sera interrompue ou refusèe, et un message d'erreur appropriè sera èmis.

Les valeurs courantes d'utilisation d'un utilisateur peuvent être remises á zèro avec la commande `GRANT`, et n'importe laquelle des clauses ci-dessus, y compris une commande `GRANT` avec les valeurs courantes.

De plus, les valeurs courantes de tous les utilisateurs peuvent être remises á zèro si les droits sont rechargès (dans le serveur, ou avec la commande `mysqladmin reload`) ou si la commande SQL `FLUSH USER_RESOURCES` est exècutèe.

Cette fonctionnalitè est activèe aussitôt qu'une limite est donnèe á un utilisateur, avec la commande `GRANT`.

Comme prè-requis á l'activation de cette fonctionnalitè, la table `user` de la base `mysql` doit contenir les colonnes additionnelles, dèfinies dans le script de crèation `mysql_install_db` et `mysql_install_db.sh`, du dossier '`scripts`'.

4.3.7 Configurer les mots de passe

Dans la plupart des cas, vous devez utiliser la commande `GRANT` pour configurer les utilisateurs et leur mot de passe. Les explications suivantes s'adressent donc aux utilisateurs avancès. Voir Section 4.3.1 [`GRANT`], page 226.

Les exemples des sections prècèdentes illustrent un principe important : lorsque vous stockez un mot de passe important avec les commandes `INSERT` ou `UPDATE`, vous devez utiliser la fonction `PASSWORD()` pour le chiffrer. Ceci est dû au fait que la table `user` stocke les mots de passe sous une forme chiffrèe, et non pas en texte clair. Si vous oubliez cela, vous risquez de configurer vos mots de passe comme ceci :

```
shell> mysql -u root mysql
mysql> INSERT INTO user (Host,User,Password)
    -> VALUES('%','jeffrey','biscuit');
mysql> FLUSH PRIVILEGES;
```

Le rèsultat est que la valeur `'biscuit'` est stockèe dans la colonne de mot de passe de la table `user`. Lorsque l'utilisateur `jeffrey` tente de se connecter au serveur avec ce mot de passe, le client `mysql` chiffre ce mot de passe avec la fonction `PASSWORD()`, gènére un vecteur d'identification basè sur la version **chiffrèe** du mot de passe et un nombre alèatoire, obtenu du serveur, puis envoie le tout au serveur. Le serveur utilise la valeur du mot de passe `password` dans la table `user` (ce qui **n'est pas la valeur chiffrèe** de `'biscuit'`) pour effectuer la recherche et comparer les rèsultats. La comparaison èchoue et le serveur rejette la connexion :

```
shell> mysql -u jeffrey -pbiscuit test
Access denied
```

Les mots de passe doivent donc être chiffrès lors de leur insertion dans la table `user`. Aussi les commandes `INSERT` doivent être spècifièes comme ceci :

```
mysql> INSERT INTO user (Host,User,Password)
    -> VALUES('%','jeffrey',PASSWORD('biscuit'));
```

Vous devez aussi utiliser la fonction `PASSWORD()` lorsque vous utilisez la commande `SET PASSWORD` :

```
mysql> SET PASSWORD FOR jeffrey@"%" = PASSWORD('biscuit');
```

Si vous modifiez les mots de passe en utilisant la commande `GRANT ... IDENTIFIED BY` ou la commande `mysqladmin password`, la fonction `PASSWORD()` n'est pas nècessaire. Ces commandes assureront le chiffrement de votre mot de passe pour vous, ce qui vous permet de spècifier le mot de passe de `'biscuit'` comme ceci :

```
mysql> GRANT USAGE ON *.* TO jeffrey@"%" IDENTIFIED BY 'biscuit';
```

or

```
shell> mysqladmin -u jeffrey password biscuit
```

Note : `PASSWORD()` n'effectue pas le chiffrement du mot de passe de la même façon qu'Unix. Il ne faut pas supposer que si vos mots de passe Unix et MySQL sont les mêmes, la fonction `PASSWORD()` retournera la même valeur que celle qui est stockèe dans le fichier de mots de passe d'UNIX. Voir Section 4.3.2 [User names], page 231.

4.3.8 Garder vos mots de passe en lieu sûr

Il est recommandè de ne pas placer votre mot de passe lá oú il risque d'être dècouvert par d'autres personnes. Les mèthode que vous utiliserez pour spècifier votre mot de passe lors de la connexion avec le client sont listèes ici, avec les risques liès á chaque mèthode :

- Ne donnez jamais l'accés á un utilisateur normal, á la table `mysql.user`. En connaissant le mot de passe chiffrè d'un utilisateur lui permettrait de se connecter sous cet utilisateur. Les mots de passe sont chiffrès pour que personne ne puisse lire le mot de passe en clair (ce qui arriverait si vous utilisiez le même mot de passe dans une autre application).

- Utilisez l'option `-pyour_pass` ou `--password=your_pass` en ligne de commande. C'est trés pratique mais trés peu sûr, car votre mot de passe devient visible aux programmes qui effectuent des bilans de l'ètat du serveur, comme la commande `ps`, qui peut être invoquèe par d'autres utilisateurs pour lister les commandes qui sont exècutèes : les clients MySQL remplacent la valeur de cet argument par une sèrie de zèro durant l'initialisation, mais il reste tout de même une courte pèriode oú ce mot de passe est lisible en clair.
- Utilisez l'option `-p` ou `--password` (sans la valeur du `mot de passe`). Dans ce cas, le programme client va solliciter la saisie du mot de passe depuis le terminal :

  ```
  shell> mysql -u user_name -p
  Enter password: ********
  ```

  Les caractéres '`*`' reprèsentent votre mot de passe.

  Cette mèthode est bien plus sûre pour saisir votre mot de passe qu'en le spècifiant directement en ligne de commande, car il n'est pas visible des autres utilisateurs. Cependant, cette mèthode n'est possible qu'avec les programmes que vous utilisez en mode interactif. Si vous voulez invoquer le client depuis un script qui s'exècute de maniére non interactive, il n'y aura pas d'opportunitè pour saisir ce mot de passe dans le terminal. Sur certains systémes, vous pourriez même voir la premiére ligne de votre script lue et interprètèe comme votre mot de passe, incorrectement.
- Stockez votre mot de passe dans le fichier de configuration. Par exemple, vous pouvez lister votre mot de passe dans la section `[client]` du fichier '`.my.cnf`' dans votre dossier personnel :

  ```
  [client]
  password=mot_de_passe
  ```

  Si vous stockez ce mot de passe dans le fichier '`.my.cnf`', le fichier ne doit pas être lisible par le groupe ou par les autres utilisateurs, ou encore accessible en ècriture : seul le propriètaire de ce fichier doit avoir ces droits. Assurez-vous les droits d'accés au fichiers sont `400` ou `600`.

  Voir Section 4.1.2 [Option files], page 198.
- Vous pouvez stocker votre mot de passe dans la variable d'environnement `MYSQL_PWD`, mais cette mèthode doit être considèrèe comme extrêmement peu sûre, et doit être èvitèe autant que possible. Certaines versions de la commande en ligne `ps` incluent une option pour afficher les variables d'environnement des processus : votre mot de passe sera alors facilement accessible, et en texte clair, si vous configurez la commande `MYSQL_PWD`. Même sur les systémes sans une telle version de la commande `ps`, il est peu recommandè de supposer que les variables d'environnement sont inaccessibles par une mèthode quelconque. Voir Annexe F [Environment variables], page 828.

En conclusion, la mèthode la plus sûre est encore de laisser le client vous demander le mot de passe, ou de le spècifier dans le fichier de configuration '`.my.cnf`'.

4.3.9 Utilisation des connexions sècurisèes

4.3.9.1 Introduction aux connexions sècurisèes

Disponible depuis la version 4.0.0, MySQL supporte les connexions sècurisèes. Pour comprendre comment MySQL utilise SSL, il est nècessaire de comprendre les concepts SSL et X509 de base. Ceux qui les connaissent, peuvent aisèment sauter ce chapitre.

Par dèfaut, MySQL utilise une connexion en clair entre le client et le serveur. Cela signifie qu'une personne peut surveiller votre trafic, et lire les donnèes èchangèes. Cette personne pourrait aussi modifier les donnèes qui transitent entre le client et le serveur. Parfois, vous aurez besoin d'èchanger des informations sur un rèseau public, mais en sècurisant ces informations. Dans ce cas, utiliser une connexion sans protection est inacceptable.

SSL est un protocole qui utilise diffèrents algorithmes de chiffrement pour s'assurer que les donnèes qui transitent par un rèseau public peuvent être considèrèes comme fiables. Ce protocole dispose de mèthodes pour s'assurer que les donnèes n'ont pas ètè modifièes, ce que soit par une altèration, une perte ou une rèpètition des donnèes. SSL inclut aussi des algorithmes pour reconnaître et fournit des outils de vèrifications d'identitèe, pris en charge par le standard X509.

Le chiffrement est une mèthode pour rendre des donnèes illisibles. En fait, les pratiques actuelles requiérent d'autres èlèments de sècuritè issus des algorithmes de chiffrement. Ils doivent savoir rèsister á de nombreux types d'attaque, comme la modification de l'ordre des messages ou les rèpètitions inopinèes.

X509 est un standard qui rend possible l'identification d'une personne sur l'internet. Il est particuliérement utilisè pour les applications e-commerce. En termes simples, il doit y avoir une entreprise (appelèe l'"autoritè de certification") qui assigne un certificat èlectronique á toute personne qui en a besoin. Ces certificats utilisent un chiffrement assymètrique qui exploitent deux clès de chiffrement, une clè publique et une clè privèe. Le propriètaire d'un certificat peut prouver son identitè en montrant son certificat á l'autre partie. Un certificat est constituè de la clè publique du propriètaire. Toute donnèe qui est chiffrèe avec cette clè publique doit être dèchiffrèe avec la clè secréte correspondante, qui est dètenue par le propriètaire du certificat.

MySQL n'utilise pas les connexions chiffrèes par dèfaut, car cela ralentit considèrablement le protocole de communication. Toute fonctionnalitè supplèmentaire requiert du travail supplèmentaire de la part du serveur, et chiffrer des donnèes est une tâche particuliérement coûteuse, qui peut ralentir considèrablement les tâches principales de MySQL. Par dèfaut, MySQL est paramètrè pour être aussi rapide que possible.

Si vous avez besoin de plus d'informations sur SSL, X509 ou le chiffrement, utilisez votre moteur de recherche prèfèrè sur Internet, et utilisez ces mots clès pour avoir plus de dètails.

4.3.9.2 Prè requis aux connexions sècurisèes

Pour faire fonctionner les connexions sècurisèes avec MySQL, vous devez disposer de ceci :

1. Installation de la librairie d'OpenSSL. Nous avons testè MySQL avec OpenSSL 0.9.6. `http://www.openssl.org/`.
2. Configuration de MySQL avec les options `--with-vio --with-openssl`.
3. Si vous utilisez une ancienne version de MySQL, vous devez modifier votre table `mysql.user`, en lui ajoutant les colonnes reliées aux notions SSL. Vous pouvez faire

cela en exècutant le script `mysql_fix_privilege_tables.sh`. Ceci est nècessaire si vos tables de droits proviennent d'un version de MySQL antèrieure á la version 4.0.0.

4. Vous pouvez vèrifier que vous possèder un serveur `mysqld` qui supporte OpenSSL en examinant le rèsultat de la commande `SHOW VARIABLES LIKE 'have_openssl'` : elle doit retourner `YES`.

4.3.9.3 Options de `GRANT`

MySQL peut vèrifier les certificats X509 en plus de la combinaisons habituelle de nom d'utilisateur et mot de passe. Toutes les options habituelles sont toujours nècessaires (nom d'utilisateur, masque d'adresse IP, nom de base de donnèes, nom de table).

Voici diffèrentes possibilitès pour limiter les connexions :

- Sans aucune option SSL ou X509, toutes les connexions chiffrèes ou non chiffrèes sont autorisèes si le nom d'utilisateur et le mot de passe sont valides.
- L'option `REQUIRE SSL` requiert que les connexions soient chiffrèes avec SSL. Notez que cette option peut être omise si il n'y a pas de ligne ACL qui autorise une connexion sans SSL.

```
mysql> GRANT ALL PRIVILEGES ON test.* TO root@localhost
    -> IDENTIFIED BY "goodsecret" REQUIRE SSL;
```

- `REQUIRE X509` impose au client d'avoir un certificat valide, mais le certificat lui même est de peu d'importance. La seule restriction est qu'il doit être possible de vèrifier la signature avec une des autoritès de certification.

```
mysql> GRANT ALL PRIVILEGES ON test.* TO root@localhost
    -> IDENTIFIED BY "goodsecret" REQUIRE X509;
```

- `REQUIRE ISSUER "issuer"` restreint les tentatives de connexion : le client doit se prèsenter avec un certificat X509 valide, èmis par l' autoritè de certification `"issuer"`. Utiliser un certificat X509 implique obligatoirement des chiffrements, donc l'option `SSL` est sous entendue.

```
mysql> GRANT ALL PRIVILEGES ON test.* TO root@localhost
    -> IDENTIFIED BY "goodsecret"
    -> REQUIRE ISSUER "C=FI, ST=Some-State, L=Helsinki,
    "> O=MySQL Finland AB, CN=Tonu Samuel/Email=tonu@mysql.com";
```

- `REQUIRE SUBJECT "subject"` impose au client d'avoir un certificat X509 valide, avec le sujet `"subject"`. Si le client prèsente un certificat valide, mais que le `"subject"` est diffèrent, la connexion est refusèe.

```
mysql> GRANT ALL PRIVILEGES ON test.* TO root@localhost
    -> IDENTIFIED BY "goodsecret"
    -> REQUIRE SUBJECT "C=EE, ST=Some-State, L=Tallinn,
    "> O=MySQL demo client certificate,
    "> CN=Tonu Samuel/Email=tonu@mysql.com";
```

- `REQUIRE CIPHER "cipher"` est utilisè pour s'assurer que les chiffrements sont suffisamment robuste, et que la bonne longueur de clè est utilisèe. SSL lui même peut être faible si des algorithmes sont utilisès avec des clès courtes. En utilisant cette option, il est possible d'imposer la mèthode de chiffrement avec la connexion.

```
mysql> GRANT ALL PRIVILEGES ON test.* TO root@localhost
    -> IDENTIFIED BY "goodsecret"
    -> REQUIRE CIPHER "EDH-RSA-DES-CBC3-SHA";
```

Les options `SUBJECT`, `ISSUER` et `CIPHER` peuvent être combinèes avec la clause `REQUIRE` comme ceci :

```
mysql> GRANT ALL PRIVILEGES ON test.* TO root@localhost
    -> IDENTIFIED BY "goodsecret"
    -> REQUIRE SUBJECT "C=EE, ST=Some-State, L=Tallinn,
    "> O=MySQL demo client certificate,
    "> CN=Tonu Samuel/Email=tonu@mysql.com"
    -> AND ISSUER "C=FI, ST=Some-State, L=Helsinki,
    "> O=MySQL Finland AB, CN=Tonu Samuel/Email=tonu@mysql.com"
    -> AND CIPHER "EDH-RSA-DES-CBC3-SHA";
```

Depuis MySQL 4.0.4, le mot clè `AND` est optionnel, entre les options `REQUIRE`.

L'ordre de ces options n'a pas d'importance, mais aucune option ne peut être spècifièe deux fois.

4.4 Prèvention des dèsastres et restauration

4.4.1 Sauvegardes de base de donnèes

Comme les tables MySQL sont stockèes sous forme de fichiers, il est facile d'en faire une sauvegarde. Pour avoir une sauvegarde consistante, faites un `LOCK TABLES` sur les tables concernèes suivi d'un `FLUSH TABLES` pour celles-ci. Voir Section 6.7.2 [`LOCK TABLES`], page 519. Voir Section 4.5.3 [`FLUSH`], page 265. Vous n'avez besoin que d'un verrou en lecture; cela permet aux autre threads de continuer á effectuer des requêtes sur les tables dont vous faites la copie des fichiers dans le dossier des bases de donnèes. `FLUSH TABLE` est requise pour s'assurer que toutes les pages d'index actifs soient ècrits sur le disque avant de commencer la sauvegarde.

Si vous voulez faire une sauvegarde d'une table avec SQL, vous pouvez utiliser `SELECT INTO OUTFILE` ou `BACKUP TABLE`. Voir Section 6.4.1 [SELECT], page 481. Voir Section 4.4.2 [BACKUP TABLE], page 243.

Une autre façon de sauvegarder une base de donnèes est d'utiliser l'utilitaire `mysqldump` ou le `script mysqlhotcopy`. Voir Section 4.8.5 [`mysqldump`], page 318. Voir Section 4.8.6 [`mysqlhotcopy`], page 322.

1. Effectuez une sauvegarde compléte de votre base de donnèes :

   ```
   shell> mysqldump --tab=/chemin/vers/un/dossier --opt --all

   ou

   shell> mysqlhotcopy base /chemin/vers/un/dossier
   ```

   Vous pouvez aussi copier tout simplement tous les fichiers de tables (les fichiers '`*.frm`', '`*.MYD`', et '`*.MYI`') du moment que le serveur ne met rien á jour. Le script `mysqlhotcopy` utilise cette mèthode.

2. Arrêtez `mysqld` si il est en marche, puis dèmarrez le avec l'option `--log-update[=nom_fichier]`. Voir Section 4.9.3 [Update log], page 328. Le ou les fichiers de log fournissent les informations dont vous avez besoin pour rèpliquer les modifications de la base de donnèes qui sont subsèquents au moment oú vous avez exècutè `mysqldump`.

Si vous avez besoin de restaurer quelque chose, essayez d'abord de recouvrer vos tables avec `REPAIR TABLE` ou `myisamchk -r` en premier. Cela devrait fonctionner dans 99.9% des cas. Si `myisamchk` ne rèussi pas, essayez la procèdure suivante (cela ne fonctionnera que si vous avez dèmarrè MySQL avec `--log-update`, voir Section 4.9.3 [Update log], page 328) :

1. Restorez le backup original de `mysqldump`.
2. Exècutez la commande suivante pour remettre en marche les mises á jours dans le log binaire :

```
shell> mysqlbinlog hostname-bin.[0-9]* | mysql
```

   Si vous utilisez le journal des mises á jour vous pouvez utiliser :

```
shell> ls -1 -t -r hostname.[0-9]* | xargs cat | mysql
```

`ls` est utilisèe pour avoir tous les fichiers de mise á jour dans le bon ordre.

Vous pouvez aussi faire des sauvegardes sèlectives avec `SELECT * INTO OUTFILE 'nom_fichier' FROM nom_de_table` et restaurez avec `LOAD DATA INFILE 'nom_fichier' REPLACE ...` Pour èviter les lignes dupliquèes, vous aurez besoin d'une `PRIMARY KEY` ou une clef `UNIQUE` dans la table. Le mot clef `REPLACE` fait que les anciens enregistrements sont remplacès par les nouveaux lorsque l'un d'eux duplique un ancien sur une valeur de clef unique.

Si vous obtenez des problémes de performances sur votre systéme, vous pouvez les contourner en mettant en place une rèplication et faisant les copies sur l'esclave au lieu du maître. Voir Section 4.10.1 [Replication Intro], page 332.

Si vous utilisez un systéme de fichiers Veritas , vous pourrez faire :

1. A partir d'un client (ou de Perl), exècutez : `FLUSH TABLES WITH READ LOCK`.
2. A partir d'un autre shell, exècutez : `mount vxfs snapshot`.
3. Depuis le premier client, exècutez : `UNLOCK TABLES`.
4. Copiez les fichiers á partir de snapshot.
5. Dèmontez snapshot.

4.4.2 Syntaxe de `BACKUP TABLE`

```
BACKUP TABLE nom_de_table[,nom_de_table...] TO '/chemin/vers/le/dossier/de/sauvegar
```

Cette commande copie le nombre minimal de fichiers de table dont en a besoin pour la restaurer vers le dossier de sauvegardes aprés avoir rafraîchit les changements dans le disque. Cela ne fonctione actuellement que pour les tables au format `MyISAM`. Pour les tables `MyISAM`, elle ne copie que les fichiers '`.frm`' (dèfinition) et '`.MYD`' (donnèes), le fichier d'indes pouvant, lui, être reconstruit á partir des deux autres.

Avant d'utiliser cette commande, merci de lire Section 4.4.1 [Backup], page 242.

Pendant la sauvegarde, un verrou de lecture est posè sur chaque table, une par une, lors de leur copie. Si vous voulez sauvegarder une image instantanèe de plusieurs table, vous

devez d'abord exècuter un `LOCK TABLES` obtenant un verrou de lecture pour chaque table concernèe.

La commande retourne une table avec les colonnes suivantes :

Colonne	Valeur
Table	Nom de la table
Op	Toujours “backup”
Msg_type	`status`, `error`, `info` ou encore `warning`.
Msg_text	Le message.

Notez que `BACKUP TABLE` n'est disponible en MySQL que depuis la version 3.23.25.

4.4.3 Syntaxe de `RESTORE TABLE`

```
RESTORE TABLE nom_de_table[,nom_de_table...] FROM '/chemin/vers/le/dossier/de/sauve
```

Restaure la ou les tables á partir d'une sauvegarde effectuèe avec `BACKUP TABLE`. Les tables existantes ne seront pas ècrasèes et dans ce cas lá, vous obtiendrez une erreur. La restauration prendra plus de temps que la sauvegarde á cause de la reconstruction du fichier d'index. Plus vous avez de clefs, plus la restauration sera longue. Just as `BACKUP TABLE`, `RESTORE TABLE` currently works only for `MyISAM` tables.

Cette commande retourne un tableau avec les colonnes suivantes :

Colonne	Valeur
Table	Nom de la table
Op	Toujours “restore”
Msg_type	`status`, `error`, `info` ou encore `warning`.
Msg_text	Le message.

4.4.4 Syntaxe de `CHECK TABLE`

```
CHECK TABLE tbl_name[,tbl_name...] [option [option...]]

option = QUICK | FAST | MEDIUM | EXTENDED | CHANGED
```

`CHECK TABLE` ne fonctionne qu'avec les tables `MyISAM` et `InnoDB`. Avec les tables `MyISAM`, c'est l'èquivalent de la commande `myisamchk -m table_name` sur la table.

Par dèfaut, l'option `MEDIUM` est utilisèe.

Cette commande vèrifie l'intègritè des tables. Pour les tables `MyISAM`, des statistiques importantes sont mises á jour. La commande retourne les infomrations suivantes sur la table dans les colonnes suivantes :

Colonne	Valeur
Table	Nom de la table.
Op	Toujours “check”.
Msg_type	Un des statut `status`, `error`, `info` ou `warning`.
Msg_text	Le message.

Notez que vous pouvez obtenir de nombreuses lignes d'informations pour chaque table. La derniére ligne sera du type `Msg_type status` et doit être normalement `OK`. Si vous n'obtenez pas de statut `OK` ou `Not checked`, il vous faudra exècuter une rèparation de la table. Voir Section 4.4.6 [Table maintenance], page 247. `Not checked` signifie que la table a indiquè qu'il n'y a pas de vèrification á faire.

Les diffèrents types de vèrifications sont les suivants :

Type	**Signification**
`QUICK`	N'analyse pas les lignes pour vèrifier les liens erronès.
`FAST`	Ne vèrifie que les tables qui n'ont pas ètè correctement fermèes.
`CHANGED`	Ne vèrifie que les tables qui ont changèes depuis la derniére vèrification, ou bien qui n'ont pas ètè correctement fermèes.
`MEDIUM`	Analyse les lignes pour s'assurer que les liens effacès sont corrects. Cette option calcule aussi la somme de contrôle des lignes, et la vèrifie avec la somme de contrôle des clès.
`EXTENDED`	Fait une vèrification compléte des liens pour chaque ligne. Cela vèrifie que la table est totalement cohèrente, mais cela peut prendre beaucoup de temps.

Pour les tables á format de dynamique de type `MyISAM`, une vèrifiation de table sera toujours dèmarrèe avec une option de niveau `MEDIUM`. Pour les tables á format de ligne statique, nous èvitons les niveaux de `QUICK` et `FAST` car les lignes sont rarement corrompues.

Vous pouvez combiner les options de vèrification comme ceci :

```
CHECK TABLE test_table FAST QUICK;
```

L'exemple ci-dessus va simplement faire une vèrification de la table, pour s'assurer qu'elle a ètè correctement fermèe.

Note : dans certains cas, `CHECK TABLE` va modifier la table! Cela arrive si la table a ètè marquèe comme 'corrupted' et 'not closed properly' mais `CHECK TABLE` n'a trouvè aucun probléme dans la table. Dans ce cas, `CHECK TABLE` va marquer la table comme correcte.

Si une table est corrompue, il est probable que les problémes sont dans les fichiers d'index et non pas dans les donnèes. Tous les types de vèrifications prèsentès ci-dessus vèrifient les index soigneusement, et ils devraient trouver la plupart des erreurs.

Si vous voulez simplement vèrifier une table que vous supposez correcte, vous pouvez n'utiliser aucune option, ou l'option `QUICK`. Cette derniére peut aussi être utilisèe si vous êtes pressè, et que vous pouvez prendre le risque minime que `QUICK` ne trouve pas d'erreur dans votre fichier. Dans la plupart des cas, MySQL doit trouver toutes les erreurs de donnèes, pour un usage normal. Si cela arrive, alors la table est marquèe comme 'corrupted', auquel cas, la table ne pourra pas être utilisèe tant qu'elle n'a pas ètè rèparèe).

`FAST` et `CHANGED` sont surtout destinèes á être utilisèes depuis un script : par exemple, il peut être exècutè depuis une tâche `cron`, si vous voulez vèrifier la table de temps en temps. Dans la plupart des cas, l'option `FAST` doit être prèfèrèe á `CHANGED` : le seul cas ou vous pourriez prèferez `CHANGED` est lorsque vous soupsonnez avoir trouvè un bogue dans les tables `MyISAM`.

`EXTENDED` ne doit être utilisè qu'aprés une vèrification normale, et que vous obtenez toujours des erreurs ètranges lorsque MySQL essaie de modifier une ligne ou trouve une ligne avec clè (ce qui est trés rare, si une vèrification a rèussie).

Certains problémes rapportès par la commande `CHECK TABLE`, ne peuvent être corrigès automatiquement :

- `Found row where the auto_increment column has the value 0`.

  Cela signifie que vous avez dans votre table une ligne qui contient la valeur 0 alors qu'elle est de type `AUTO_INCREMENT`. (Il est possible de crèer une ligne oú la colonne `AUTO_INCREMENT` vaut 0 en spècifiant explicitement la valeur 0 dans la colonne avec la commande `UPDATE`.

  Ce n'est pas une erreur en soit, mais cela peut poser des problémes si vous dècidez de sauver cette table dans un fichier texte, et de la restaurer, ou encore d'appliquer la commande `ALTER TABLE` sur la table. Dans ce cas, la colonne `AUTO_INCREMENT` va changer automatiquement de valeur, en suivant les régles des colonnes de type `AUTO_INCREMENT`, qui vont causer un probléme de clè doublon.

  Pour se dèbarasser de cette alerte, vous devez utiliser une commande `UPDATE` surla table, pour mettre une valeur diffèrente de 0 dans cette colonne.

4.4.5 Syntaxe de `REPAIR TABLE`

```
REPAIR TABLE tbl_name[,tbl_name...] [QUICK] [EXTENDED] [USE_FRM]
```

`REPAIR TABLE` ne fonctionne que les tables de type `MyISAM`, et c'est l'èquivalent de la commande en ligne `myisamchk -r table_name`.

Normalement, vous n'avez pas á exècuter cette commande, mais si une catastrophe vous frappe, vous êtes presque assurès de retrouver vos donnèes dans les tables MyISAM, avec la commande `REPAIR TABLE`. Si vos tables sont souvent corrompues, vous devrie toutefois rechercher la cause de ce probléme! Voir Section A.4.1 [Crashing], page 697. Voir Section 7.1.3 [MyISAM table problems], page 537.

`REPAIR TABLE` rèpare autant que possible les tables corrompues. La commande retourne la table suivante :

Colonne	**Valeur**
Table	Nom de la table
Op	Toujours “repair”
Msg_type	Un des statut `status`, `error`, `info` ou `warning`.
Msg_text	Le message.

Notez que vous pourriez obtenir de nombreux messages pour chaque table. La derniére ligne doit être du format `Msg_type status` et doit être normalement `OK`. Si vous n'obtenez pas `OK`, vous devez essayer de rèparer votre table avec la commande `myisamchk -o`, car `REPAIR TABLE` de supporte pas encore toutes les options de `myisamchk`. Dans un futur proche, nous allons rendre cette commande encore plus souple.

Si l'option `QUICK` est fournie, alors MySQL va essayer de ne rèparer que le fichier d'index.

Si vous utilisez l'option `EXTENDED`, alors MySQL va essayer de crèer l'index ligne par ligne, au lieu de crèer un index á la fois, par tri. C'est une mèthode qui peut s'avèrer plus efficace que de trier sur des clès de taille fixe, si vous avez des clès `CHAR` longues qui se compressent bien. Ce type de rèparation est l'èquivalent de `myisamchk --safe-recover`.

Depuis `MySQL` 4.0.2, il existe le mode `USE_FRM` pour `REPAIR`. Utilisez-le si le fichier ‘`.MYI`’ manque, ou si son entête est corrompu. Avec ce mode, MySQL va recrèer la table, en

utilisant les informations dans le fichier '`.frm`'. Ce type de rèparation ne peut pas être fait avec `myisamchk`.

4.4.6 Utilisation de `myisamchk` pour la maintenance des tables et leur recouvrement

A partir de la version 3.23.13 de MySQL, vous pouvez vèrifier vos tables MyISAM avec la commande `CHECK TABLE`. Voir Section 4.4.4 [CHECK TABLE], page 244. Vous pouvez les rèparer avec la commande `REPAIR TABLE`. Voir Section 4.4.5 [REPAIR TABLE], page 246.

Pour vèrifier/rèparer les tables MyISAM ('`.MYI`' et '`.MYD`') vous devez utiliser l'utilitaire `myisamchk`. Pour vèrifier/rèparer les tables ISAM ('`.ISM`' et '`.ISD`') vous devez utiliser l'utilitaire `isamchk`. Voir Chapitre 7 [Table types], page 531.

Dans ce qui suit, nous allons parler de `myisamchk`, mais tout s'applique aussi á l'ancien `isamchk`.

Vous pouvez utiliser l'utilitaire `myisamchk` pour obtenir des informations á propos des tables de la base de donnèes, les vèrifier et rèparer, ou les optimiser. La section suivante dècrit comment invoquer `myisamchk` (incluant une description de ses options), comment mettre en place un planificateur de maintenance, et comment utiliser les diffèrentes fonctionnalitès de `myisamchk`.

Vous pouvez, dans la plupart des cas, utiliser la commande `OPTIMIZE TABLES` pour optimiser et rèparer les tables, mais ce n'est ni aussi rapide, ni aussi sûr (en cas d'erreurs fatales rèelles) que `myisamchk`. D'un autre côtè, `OPTIMIZE TABLE` est plus facile á utiliser et vous n'avez pas besoin de vous occuper de libèrer les tables. Voir Section 4.5.1 [`OPTIMIZE TABLE`], page 264.

Même si la rèparation de `myisamchk` est assez sècurisèe, il est toujours bon de crèer une sauvegarde *avant* d'effectuer les rèparations (ou quoi que ce soit qui puisse entraîner de grosses modifications sur la table)

4.4.6.1 Syntaxe de l'utilitaire `myisamchk`

`myisamchk` s'exècute avec une commande de la forme :

```
shell> myisamchk [options] tbl_name
```

Les `options` spècifient ce que vous voulez que `myisamchk` fasse. Elles sont dècrites dans ce chapitre. Vous pouvez aussi obtenir une liste d'options en invoquant le programme avec `myisamchk --help`. Sans option, `myisamchk` va simplement vèrifier les tables. Pour obtenir plus d'information ou pour demander á `myisamchk` de prendre des mesures correctives, il faut ajouter l'une des options listèes ici.

`tbl_name` est la table que vous voulez rèparer ou vèrifier. Si vous exècutez `myisamchk` autre part que dans le dossier de donnèes, vous devez spècifier le chemin jusqu'au fichier, car sinon, `myisamchk` n'aura aucune idèe d'oú chercher les donnèes dans votre base. En fait, `myisamchk` ne se prèoccupe pas du fait que le fichier que vous utilisez est dans le dossier de base ou pas : vous pouvez copier le fichier á rèparer dans un autre dossier, et y faire les opèrations d'entretien.

Vous pouvez spècifier plusieurs noms de tables á `myisamchk` si vous le voulez. Vous pouvez aussi spècifier un nom sous la forme d'un fichier d'index (avec l'option '`.MYI`'), qui vous

permettra de spècifier toutes les tables dans un dossier en utilisant le schèma '`*.MYI`'. Par exemple, si vous êtes dans le dossier de donnèes, vous pouvez spècifier toutes les tables dans le dossier comme ceci :

```
shell> myisamchk *.MYI
```

Si vous n'êtes pas dans le dossier de donnèes, et que vous souhaitez vèrifier toutes les tables, vous devez ajouter le chemin jusqu'au dossier :

```
shell> myisamchk /path/to/database_dir/*.MYI
```

Vous pouvez même vèrifier toutes les tables de toutes les bases avec le chemin suivant :

```
shell> myisamchk /path/to/datadir/*/*.MYI
```

La mèthode recommandèe pour vèrifier rapidement toutes les tables est :

```
myisamchk --silent --fast /path/to/datadir/*/*.MYI
isamchk --silent /path/to/datadir/*/*.ISM
```

Si vous voulez vèrifier toutes les tables et rèparer celles qui sont corrompues, vous pouvez utiliser la ligne suivante :

```
myisamchk --silent --force --fast --update-state -O key_buffer=64M \
          -O sort_buffer=64M -O read_buffer=1M -O write_buffer=1M \
          /path/to/datadir/*/*.MYI
isamchk --silent --force -O key_buffer=64M -O sort_buffer=64M \
        -O read_buffer=1M -O write_buffer=1M /path/to/datadir/*/*.ISM
```

La commande ci-dessus suppose que vous avez plus de 64 Mo de libres.

Notez que si vous obtenez une erreur comme celle-ci :

```
myisamchk: warning: 1 clients is using or hasn't closed the table properly
```

Cela signifie que vous essayez de vèrifier une table qui a ètè modifièe par un autre programme (comme le serveur `mysqld`) qui n'a pas encore refermè le fichier de table, ou que le fichier n'a pas ètè correctement refermè.

Si `mysqld` fonctionne, vous devez forcer la fermeture correcte des fichiers de tables avec la commande `FLUSH TABLES`, et vous assurer que personne n'utilise les tables durant vos opèrations avec `myisamchk`. En MySQL version 3.23, la meilleure mèthode pour èviter ce probléme est d'utiliser la commande `CHECK TABLE` au lieu de `myisamchk` pour vèrifier les tables.

4.4.6.2 Options gènèrales de `myisamchk`

`myisamchk` supporte les options suivantes :

`-# or --debug=debug_options`
: Affiche le log de dèbogage. La chaîne `debug_options` vaut souvent : `'d:t:o,filename'`.

`-? or --help`
: Affiche le message d'aide, et termine le programme.

`-O var=option, --set-variable var=option`
: Spècifie la valeur d'une variable. Notez bien que `--set-variable` est obsoléte depuis MySQL 4.0, il suffit dèsormais d'utiliser `--var=option`. Les vari-

ables disponibles et leur valeurs par dèfaut sont accessibles avec la commande `myisamchk --help`:

Variable	Valeur
key_buffer_size	523264
read_buffer_size	262136
write_buffer_size	262136
sort_buffer_size	2097144
sort_key_blocks	16
decode_bits	9

`sort_buffer_size` est utilisèe pour la rèparation de clès, en les triant, ce qui est le cas par dèfaut avec l'option `--recover`.

`key_buffer_size` est utilisèe lorsque vous vèrifiez une table avec l'option `--extended-check` ou lorsque les clès sont rèparèes par insertion des clès ligne par ligne (comme lors des insertions normales). La rèparation via le buffer de clè est utilisèe dans les situations suivantes :

- Si vous utilisez l'option `--safe-recover`.
- Si les fichiers temporaires oú il faut trier les clès, seront deux fois plus grands que lors de la crèation direct du fichier de clè. C'est souvent le cas lorsque vous avez de grandes colonnes de type `CHAR`, `VARCHAR` ou `TEXT` car le tri doit stocker la valeur totale de la colonne durant l'opèration. Si vous avez beaucoup d'espace disque temporaire, et que vous pouvez forcer `myisamchk` á rèparer avec la mèthode de tri, vous pouvez utiliser l'option `--sort-recover`.

La rèparation avec la mèthode du buffer de clè prend bien moins d'espace disque que d'utiliser le tri, mais c'est aussi bien plus lent.

Si vous voulez une rèparation plus rapide, donnez á la variable ci-dessus la valeur du quart de votre mèmoire disponible. Vous pouvez donner de grandes valeurs aux deux variables, car seul un des deux tampons dècrits sera utilisè.

`-s or --silent`
: Mode silencieux. N'affiche que les erreurs. Vous pouvez utiliser deux fois cette option (`-ss`) pour rendre `myisamchk` trés silencieux.

`-v or --verbose`
: Mode dètaillè. Affiche beaucoup de dètails sur les opèrations en cours. Vous pouvez l'utiliser avec les options `-d` et `-e`. Utilisez `-v` plusieurs fois pour avoir encore plus d'informations (`-vv`, `-vvv`)!

`-V or --version`
: Affiche la version de `myisamchk` et s'arrête.

`-w or, --wait`
: Au lieu d'afficher une erreur si la table est verrouillèe, attend jusqu'á ce que la table soit libre avant de continuer. Notez que si vous utilisez `mysqld` avec l'option `--skip-external-locking`, la table ne pourra être verrouillèe que par une autre commande `myisamchk`.

4.4.6.3 Options de vèrifications pour `myisamchk`

`-c ou --check`
: Vèrifie les erreurs d'une table. Ceci est l'opèration par dèfaut de `myisamchk` si vous ne lui donnez aucune autre option.

`-e ou --extend-check`
: Vèrifie la table minutieusement (ce qui est un peu lent si vous avez des index). Cette option ne doit être utilisèe que pour les cas extrémes. Normalement, `myisamchk` ou `myisamchk --medium-check` devrait, dans la plupart des cas, être capable de trouver s'il y'a des erreurs dans la table.

  Si vous utilisez `--extended-check` et que vous avez beaucoup de mèmoire, vous devez augmenter de beaucoup la valeur de `key_buffer_size` !

`-F ou --fast`
: Ne vèrifie que les tables qui n'ont pas ètè fermèes proprement.

`-C ou --check-only-changed`
: Ne vèrifie que les tables qui ont changè depuis la derniére vèrification.

`-f ou --force`
: Redèmarrez `myisamchk` avec `-r` (rèpare) sur la table, si `myisamchk` trouve une erreur dans la table.

`-i ou --information`
: Affiche des statistiques á propos de la table vèrifièe.

`-m ou --medium-check`
: Plus rapide que extended-check, mais ne trouve que 99.99% des erreurs. Devrait, cependant, être bon pour la plupart des cas.

`-U ou --update-state`
: Enregistre dans le fichier '`.MYI`' lorsque la table á ètè vèrifièe ou a crashè. Cela devrait être utilisè pour tirer tous les avantages de l'option `--check-only-changed`, mais vous ne devez pas utiliser cette option si le serveur `mysqld` utilise cette table et que vous utilisez `mysqld` avec `--skip-external-locking`.

`-T ou --read-only`
: Ne marque pas la table comme vèrifièe. C'est pratique si vous utilisez `myisamchk` pour vèrifier une table issue d'une autre application qui n'utilise pas les verrous. (comme `mysqld --skip-external-locking`).

4.4.6.4 Options de rèparation de myisamchk

Les options suivantes sont utilisèes avec `myisamchk` et l'option de rèparation `-r` ou `-o`:

`-D # ou --data-file-length=#`
: Taille maximale du fichier de donnèes (lors de la recrèation du fichier de donnèes, et qu'il est 'complet').

`-e ou --extend-check`
: Essaie de retrouver toutes les lignes possibles du fichier de donnèes. Normalement, cette option va aussi dècouvrir beaucoup de ligne erronèes. N'utilisez pas cette option si vous n'êtes pas totalement desespèrè.

`-f ou --force`
: Ecrase les anciens fichiers temporaires (`table_name.TMD`) au lieu d'annuler.

`-k # ou keys-used=#`
: Si vous utilisez les tables ISAM, indique au gestionnaire de table ISAM qu'il doit uniquement modifier les `#` premiers index. Si vous utilisez le gestionnaire de table `MyISAM`, cette option indique quelles clès utiliser, et chaque bit binaire reprèsente une clè (la premiére clè est le bit 0). Cela permet de rèaliser des insertions plus rapides. Les index dèsactivès pourront être rèactivès avec l'option `myisamchk -r`.

`-l ou --no-symlinks`
: Ne pas suivre les lignes symboliques. Normalement, `myisamchk` rèpare les tables qu'un lien symbolique reprèsente. Cette option n'existe pas en MySQL 4.0, car MySQL 4.0 ne va pas supprimer les liens symboliques durant la rèparation.

`-r ou --recover`
: Peut rèparer presque tout, sauf les clès uniques qui ne le sont plus (ce qui est extrêmement rare avec les tables ISAM/MyISAM). Si vous voulez restaurer un table, c'est l'option á utiliser en premier. Si myisamchk indique que la table ne peut pas être corrigèe avec l'option `-r`, vous pouvez alors passer á l'option `-o`. Notez que dans le cas rarissime oú `-r`, le fichier de donnèes est toujours intact. Si vous avez beaucoup de mèmoire, vous pouvez augmenter la taille du buffer `sort_buffer_size`!

`-o ou --safe-recover`
: Utilise une ancienne mèthode de restauration (lit toutes les lignes dans l'ordre, et modifie l'arbre d'index conformèment pour les lignes trouvèes). C'est une mèthode qui est beaucoup plus lente que l'option `-r`, mais elle est capable de traiter certaines situations exceptionnelles que `-r` ne pourrait pas traiter. Cette mèthode utilise aussi moins d'espace disque que `-r`. Normalement, vous devriez commencer á rèparer avec l'option `-r`, et uniquement sur l'èchec de cette option, passer á `-o`.

  Si vous avez beaucoup de mèmoire, vous devriez augmenter la taille du buffer de clè ! `key_buffer_size`!

`-n ou --sort-recover`
: Force `myisamchk` á utiliser le tri pour rèsoudre les clès, même si le fichier temporaire doit être ènorme.

`--character-sets-dir=...`
: Dossier qui contient les jeux de caractéres.

`--set-character-set=name`
: Change le jeu de caractéres utilisè par l'index.

`-t ou --tmpdir=path`
: Chemin pour stocker les fichiers temporaires. Si cette option n'est pas fournie, `myisamchk` va utiliser la variable d'environnement `TMPDIR` pour cela.

`-q ou --quick`
: Rèparation rapide, sans modifier le fichier de donnèes. Il est possible d'ajouter l'option `-q` pour forcer `myisamchk` á modifier le fichier original en cas de clès doublons.

`-u ou --unpack`
: Dècompresse des donnèes compressèes avec myisampack.

4.4.6.5 Autres options de `myisamchk`

Les autres actions que `myisamchk` peut rèaliser, en dehors de vèrifier et rèparer une table sont :

`-a or --analyze`
: Analyser la distribution des clès. Cela amèliore les performances des jointures en permettant á l'optimiseur de jointure de mieux choisir l'ordre d'utilisation des clès. `myisamchk --describe --verbose table_name'` ou `SHOW KEYS` dans MySQL.

`-d or --description`
: Affiche des informations sur la table.

`-A or --set-auto-increment[=value]`
: Force `AUTO_INCREMENT` á commencer avec une valeur supèrieure. Si aucune valeur n'est fournie, la prochaien valeur de la colonne `AUTO_INCREMENT` sera la plus grande valeur de la colonne +1.

`-S or --sort-index`
: Trie les blocs de l'arbre d'index dans l'ordre haut-bas. Cela va optimiser les recherches, et les scans de tables par clès.

`-R or --sort-records=#`
: Trie les lignes en fonction de l'index. Cela rassemble vos donnèes, et peut accèlèrer les lectures de lignes par intervalle avec `SELECT` et `ORDER BY` sur cet index (ce tri peut être TRES lent la premiére fois). Pour connaitre les numèros d'index de tables, utilisez la commande `SHOW INDEX`, qui affiche les index dans le même ordre que `myisamchk` ne les voit. Les index sont numèrotès á partir de 1.

4.4.6.6 Utilisation de la mèmoire par `myisamchk`

L'espace mèmoire est trés important quand vous utilisez `myisamchk`. `myisamchk` n'utilise pas plus de mèmoire que ce que vous spècifiez avec les options `-O`. Si vous pensez utiliser `myisamchk` sur des fichiers trés grands, vous devez d'abord dècider la quantitè de mèmoire que vous souhaitez utiliser. Avec des valeurs plus grandes, vous pouvez accèlèrer `myisamchk`. Par exemple, si vous avez plus de 32 Mo de RAM, vous pourriez utiliser les options suivantes (en plus des autres options que vous pourriez spècifier) :

```
shell> myisamchk -O sort=16M -O key=16M -O read=1M -O write=1M ...
```

Utiliser `-O sort=16M` sera probablement suffisant pour la plupart des cas.

Soyez conscient que `myisamchk` utilise des fichiers temporaires dans le dossier `TMPDIR`. Si `TMPDIR` est un fichier en mèmoire, vous pourriez facilement rencontrer des erreurs de mèmoire. Si cela arrive, choisissez une autre valeur pour `TMPDIR`, avec plus d'espace disque, et redèmarrez `myisamchk`.

Lors de la rèparation, `myisamchk` va aussi avoir besoin d'espace disque :

- Doublez la taille du fichier de donnèes (l'original plus une copie). Cet espace n'est pas nècessaire si vous faites des rèparations de type `--quick`, car dans ce cas, seul le fichier d'index sera recrèè. Cet espace est nècessaire sur le même disque que l'original !
- De l'espace pour le nouveau fichier d'index qui remplacera l'ancien. L'ancien fichier d'index est rèduit dés le dèmarrage, ce qui vous permet gènèralement d'ignorer cet espace. Cet espace est nècessaire sur le même disque que l'original !
- Lorsque vous utilisez les options `--recover` ou `--sort-recover` (mais pas lorsque vous utilisez `--safe-recover`), vous aurez besoin d'espace pour le buffer de tri : `(plus_grande_cle + taille_du_pointeur_de_ligne)*nombre_de_lignes * 2`. Vous pouvez vèrifier la taille des clès et la taille du pointeur de ligne avec la commande `myisamchk -dv table`. Cet espace est allouè sur le disque temporaire (spècifiè par `TMPDIR` par `--tmpdir=#`).

Si vous avez des problémes avec l'espace disque durant la rèparation, vous pouvez utiliser l'option `--safe-recover` au lieu de `--recover`.

4.4.6.7 Utiliser `myisamchk` pour restaurer une table

Si vous utilisez `mysqld` avec l'option `--skip-external-locking` (qui est la configuration par dèfaut pour certains systémes, comme Linux), vous ne pouvez pas utiliser `myisamchk` pour vèrifier une table, lorsque `mysqld` utilise aussi la table. Si vous pouvez être sûr que personne n'utilise cette table via `mysqld` lorsque vous utilisez `myisamchk`, vous n'aurez qu'á utiliser la commande `mysqladmin flush-tables` avant de commencer á vèrifier les tables. Si vous ne pouvez pas garantir cette condition, vous devez alors èteindre le serveur `mysqld` pour vèrifier les tables. Si vous exècutez `myisamchk` alors que `mysqld` modifie la table, vous pourriez obtenir un diagnostic de corruption de la table, alors que ce n'est pas le cas.

Si vous n'utilisez pas l'option `--skip-external-locking`, vous pouvez vous servir de `myisamchk` pour vèrifier les tables á tout moment. Pendant que vous le faites, les autres clients qui tentent de modifier la table devront attendre que `myisamchk` ait fini.

Si vous utilisez `myisamchk` pour rèparer ou optimiser les tables, vous **devez** toujours vous assurer que `mysqld` n'utilise pas cette table (ce qui s'applique aussi si vous utilisez `--skip-external-locking`). Si vous n'èteignez pas le serveur `mysqld`, vous devez au moins utiliser `mysqladmin flush-tables` avant de lancer `myisamchk`. Vos tables **peuvent être corrompues** si le serveur et `myisamchk` travaillent dans une même table simultanèment.

Ce chapitre dècrit comment vèrifier et gèrer les corruptions de donnèes dans les bases MySQL. Si vos tables sont frèquemment corrompues, vous devriez commencer par en rechercher la raison ! Voir Section A.4.1 [Crashing], page 697.

La section sur les tables `MyISAM` contient diffèrentes raisons pour lesquelles une table peut être corrompue. Voir Section 7.1.3 [MyISAM table problems], page 537.

Lorsque vous effectuez une restauration de table, il est important que chaque table `tbl_name` dans une base corresponde aux trois fichiers dans le dossier de base, du dossier de donnèes :

Fichier	**Utilisation**
'`tbl_name.frm`'	Dèfinition de la table
'`tbl_name.MYD`'	Fichier de donnèes
'`tbl_name.MYI`'	Fichier d'index

Chacun de ces trois fichiers est sujet á des corruptions diverses, mais les problémes surviennent gènèralement dans les fichiers de donnèes ou d'index.

`myisamchk` fonctionne en crèant une copie du fichier '`.MYD`' (les donnèes), ligne par ligne. Il termine sa rèparation en supprimant l'ancien fichier '`.MYD`' et en renommant le nouveau á la place de l'ancien. Si vous utilisez l'option `--quick`, `myisamchk` ne crèe pas de fichier temporaire mais suppose plutôt que le fichier '`.MYD`' est correct et il gènére simplement un nouveau fichier d'index sans toucher au fichier '`.MYD`'. C'est une mèthode sècuritaire, car `myisamchk` va automatiquement dètecter si le fichier '`.MYD`' est corrompu, et annulera alors la rèparation si c'est le cas. Vous pouvez aussi ajouter deux options `--quick` á `myisamchk`. Dans ce cas, `myisamchk` ne s'interrompt pas sur certaines erreurs (comme des clès doublons), et essaie de rèsoudre ce probléme en modifiant le fichier '`.MYD`'. Normalement, l'utilisation de deux options `--quick` n'est utile que si vous n'avez pas trop d'espace disque pour rèaliser la rèparation. Dans ce cas, vous devez au moins faire une copie de sauvegarde avant d'utiliser `myisamchk`.

4.4.6.8 Comment vèrifier la cohèrence d'une table

Pour vèrifier les tables de type MyISAM, utilisez les commandes suivantes :

`myisamchk nom_de_table`
: Cette commande trouvera 99.99% de toutes les erreurs. Ce qu'elle ne peut pas dècouvrir comme erreurs, sont celles qui impliquent **uniquement** le fichier de donnèes (ce qui est trés inhabituel). Si vous voulez vèrifier une table, vous devriez utiliser l'utilitaire `myisamchk` sans les options ou avec les options `-s` ou `--silent`.

`myisamchk -m nom_de_table`
: Cette commande trouvera 99.999% de toutes les erreurs. Elle vèrifie toutes les entrèes dans le fichier d'index, puis lit toutes les lignes. Elle calcule une somme de contrôle pour toutes les clès et les lignes, et vèrifie que les deux se correspondent dans l'arbre d'index.

`myisamchk -e nom_de_table`
: Cette commande fait une vèrification compléte et exhaustive de toutes les donnèes (`-e` signifie "extended check"). Elle fait une lecture de contrôle de chaque ligne, pour vèrifier qu'elle correspond bien aux index. Cette commande va prendre un long moment sur les grosses tables. `myisamchk` va normalement s'arrêter dés qu'il trouve une erreur. Si vous voulez obtenir plus d'information

sur cette erreur, vous pouvez utiliser l'option `--verbose` (ou `-v`). Cela fera que `myisamchk` va continuer á travailler et accumuler jusqu'á 20 erreurs. En utilisation normale, l'utilisation de cet utilitaire sans options est suffisante.

`myisamchk -e -i nom_de_table`
Comme les commandes prècèdentes, mais l'option `-i` indique á `myisamchk` qu'il doit afficher des informations statistiques.

4.4.6.9 Comment rèparer des tables

Dans la section prèsente, nous allons uniquement parler de l'utilitaire `myisamchk` sur les tables `MyISAM` (extensions '`.MYI`' et '`.MYD`'). Si vous utilisez les tables `ISAM` (extensions '`.ISM`' et '`.ISD`'), vous devriez vous servir de `isamchk` á la place.

Depuis MySQL version 3.23.14, vous pouvez rèparer les tables MyISAM avec la commande SQL `REPAIR TABLE`. Voir Section 4.4.5 [REPAIR TABLE], page 246.

Les symptômes de corruption de tables sont des requêtes qui s'interrompent inopinèment :

- '`tbl_name.frm`' locked against change : '`tbl_name.frm`' est verrouillèe en ècriture
- Can't find file '`tbl_name.MYI`' (Errcode: ###) : Impossible de trouver le fichier '`tbl_name.MYI`' (Errcode: ###)
- Unexpected end of file : Fin de fichier inattendue
- Record file is crashed : Fichier de donnèes crashè
- Got error ### from table handler : Reception de l'erreur ### de la part du gestionnaire de table

  Pour obtenir plus d'informations sur l'erreur, vous pouvez exècuter la commande `perror ###`. Voici les erreurs les plus courantes :

```
shell> perror 126 127 132 134 135 136 141 144 145
126 = Index file is crashed / Wrong file format : le fichier d'index est corromp
127 = Record-file is crashed : le fichier de donnèes est corrompu.
132 = Old database file / ce fichier provient d'une vieille base de donnèes.
134 = Record was already deleted (or record file crashed) / La ligne ètait dèjá
135 = No more room in record file / Plus de place dans le fichier de donnèes.
136 = No more room in index file / Plus de place dans le fichier d'index.
141 = Duplicate unique key or constraint on write or update / Doublon pour une c
144 = Table is crashed and last repair failed / la table est corrompue et la der
145 = Table was marked as crashed and should be repaired / La table a ètè marquè
```

  Notez que l'erreur 135, "no more room in record file", n'est pas une erreur qui sera facile á corriger. Dans ce cas, vous devez utiliser la commande suivante :

```
ALTER TABLE table MAX_ROWS=xxx AVG_ROW_LENGTH=yyy;
```

Dans d'autres cas, vous devrez rèparer vos tables. `myisamchk` peut gènèralement dètecter et corriger la plupart des erreurs.

Le processus de rèparation se dèroule en 4 ètapes dècrites ici. Avant de vous lancer, vous devriez vous placer dans le dossier de donnèes et vèrifier les permissions des fichiers de donnèes. Assurez-vous qu'ils sont bien lisibles par l'utilisateur Unix que MySQL utilise (et

á vous aussi, car vous aurez besoin d'accèder á ces fichiers durant la vèrification. Si vous devez corriger ces fichiers, vous aurez aussi besoin des droits d'ècriture.

Si vous utilisez MySQL version 3.23.16 et plus rècent, vous pouvez (et vous devriez) utiliser les commandes `CHECK` et `REPAIR` pour rèparer vos tables `MyISAM`. Voir Section 4.4.4 [CHECK TABLE], page 244. Voir Section 4.4.5 [REPAIR TABLE], page 246.

La section du manuel sur l'entretien des tables inclut la prèsentation des options des utilitaires `isamchk/myisamchk`. Voir Section 4.4.6 [Table maintenance], page 247.

La section suivante est destinèe aux cas oú les commandes ci-dessus ont èchouè ou que vous voulez exploiter les fonctionnalitès avancèes que `isamchk/myisamchk` proposent.

Si vous allez rèparer une table en ligne de commande, il est recommandè d'arrêter le serveur `mysqld`. Notez que lorque vous exècutez une commande `mysqladmin shutdown` sur un serveur distant, le serveur `mysqld` sera encore opèrationnel pendant un instant aprés que `mysqladmin` ait terminè, jusqu'á ce que toutes les requêtes et toutes les clès aient ètè ècrites sur le disque.

Etape 1 : Vèrifier vos tables

Exècutez la commande `myisamchk *.MYI` ou `myisamchk -e *.MYI` si vous avez plus de temps. Utilisez `-s` (silencieux) pour supprimer les informations peu pertinentes.

Si le serveur `mysqld` a terminè, vous devriez utiliser l'option –update pour indiquer á `myisamchk` d'enregistrer la vèrification des tables ('checked').

Vous n'aurez á rèparer que les tables pour lesquelles `myisamchk` vous annonce une erreur. Pour de telles tables, passez á l'ètape 2.

Si vous obtenez des erreurs ètranges lors de la vèrification, (comme, l'erreur `out of memory`), ou si `myisamchk` crashe, passez á l'ètape 3.

Etape 2 : rèparation simple et facile

Note : Si vous voulez rèparer trés rapidement, vous devriez ajouter `-O sort_buffer=#` `-O key_buffer=#` (oú # vaut environ le quart de la mèmoire du serveur), á toutes les commandes `isamchk/myisamchk`.

Premiérement, essayez `myisamchk -r -q tbl_name` (`-r -q` signifie "mode de rèparation rapide"). Cette commande va tenter de rèparer le fichier d'index sans toucher au fichier de donnèes. Si le fichier de donnèes contient toutess les donnèes qu'il est sensè contenir, et que les points d'ancrage pour les effacements sont corrects, cette commande doit rèussir, et la table sera alors rèparèe. Passez alors á la table suivante. Sinon, suivez la procèdure suivante :

1. Faites une copie de sauvegarde de votre fichier de donnèes.
2. Utilisez la commande `myisamchk -r tbl_name` (`-r` signifie "mode de rèparation"). Cette commande va supprimer les lignes invalides et effacer ces lignes du fichier de donnèes, puis reconstruire le fichier d'index.
3. Si l'instruction prècèdente a èchouè, utilisez `myisamchk --safe-recover tbl_name`. Le mode restauration sècuritaire utilise une vieille mèthode de rèparation qui peut gèrer certains cas rares, mais elle est bien plus lente.

Si vous obtenez des erreurs ètranges lors de la rèpaaration (comme des erreurs de type `out of memory`), ou si `myisamchk` crashe, passez á l'ètape 3.

Etape 3 : Rèparations difficiles

Nous ne devriez atteindre cette ètape que si les 16 premiers ko du fichier d'index sont dètruits, ou qu'il contient des donnèes erronèes, ou si le fichier d'index manque. Dans ce cas, il est nècessaire de crèer un nouveau fichier d'index. Faites ceci :

1. Dèplacez le fichier de donnèes dans une archive sûre.
2. Utilisez le fichier description de la table pour crèer de nouveaux fichiers de donnèes et d'index vides.

   ```
   shell> mysql db_name
   mysql> SET AUTOCOMMIT=1;
   mysql> TRUNCATE TABLE table_name;
   mysql> quit
   ```

   Si votre version SQL ne dispose pas de `TRUNCATE TABLE`, utilisez la commande `DELETE FROM table_name`.
3. Copiez l'ancien fichier de donnèes á la place du nouveau fichier de donnèes (ne faites pas un simple dèplacement de fichier. Utilisez une copie, au cas oú un probléme surviendrait).

Retournez á l'ètape 2. `myisamchk -r -q` doit alors fonctionner (et ceci ne doit pas être une boucle infinie).

Depuis `MySQL` 4.0.2, vous pouvez aussi utiliser `REPAIR ... USE_FRM` qui effectue toute cette opèration automatiquement.

Etape 4 : Rèparation trés difficiles

Vous ne devriez atteindre cette ètape que si votre fichier de description a aussi crashè. Cela ne devrait jamais arriver, car le fichier de description n'est jamais modifiè une fois que la table est crèèe.

1. Restaurez le fichier de description avec une sauvegarde, et retournez á l'ètape 3. Vous pouvez aussi restaurer le fichier d'index et retourner á l'ètape 2. Dans ce dernier cas, vous pouvez dèmarrer avec l'option `myisamchk -r`.
2. Si vous n'avez pas de sauvegarde, mais que vous savez exactement comment la table a ètè crèèe, vous pouvez crèer une telle table dans une autre base. Supprimez alors le nouveau fichier de donnèes, puis dèplacez les fichiers de description et d'index dans votre base de donnèes crashèe. Cela vous donnera un nouveau fichier d'index et de description. Retournez á l'ètape 2 et essayez de reconstruire le fichier d'index.

4.4.6.10 Optimisation de tables

Pour rèorganiser les lignes fragmentèes et èliminer l'espace perdu par les effacements et les modifications de lignes, vous pouvez exècuter l'utilitaire `myisamchk` en mode de restauration :

```
shell> myisamchk -r tbl_name
```

Vous pouvez optimiser une table de la même façon que vous le faîtes avec la commande SQL `OPTIMIZE TABLE`. `OPTIMIZE TABLE` effectue une rèparation de la table, et une analyse des index, puis trie l'arbre d'index pour accèlèrer les recherches de clès. L'utilisation de la commande rèduit aussi les interfèrences entre le serveur et l'utilitaire car c'est le serveur lui-même qui fait le travail. Voir Section 4.5.1 [OPTIMIZE TABLE], page 264.

`myisamchk` dispose aussi d'un grand nombre d'options que vous pouvez utiliser pour amèliorer les performances de la table :

- `-S`, `--sort-index`
- `-R index_num`, `--sort-records=index_num`
- `-a`, `--analyze`

Pour une description compléte de ces options, Voir Section 4.4.6.1 [myisamchk syntax], page 247.

4.4.7 Mettre en place un règime d'entretien de MySQL

Depuis MySQL version 3.23.13, vous pouvez vèrifier les tables de type MyISAM avec la commande `CHECK TABLE`. Voir Section 4.4.4 [CHECK TABLE], page 244. Vous pouvez aussi rèparer les tables avec la commande `REPAIR TABLE`. Voir Section 4.4.5 [REPAIR TABLE], page 246.

C'est une bonne idèe que d'effectuer des vèrifications des tables règuliérement, plutôt que d'attendre qu'un probléme survienne. Pour faire ces vèrifications, vous pouvez utiliser la commande `myisamchk -s`. L'option `-s` (raccourci pour `--silent`) fait que `myisamchk` s'exècute en mode silencieux, et n'affiche que les messages d'erreurs.

C'est aussi une bonne idèe que de vèrifier les tables lorsque le serveur dèmarre. Par exemple, á chaque fois qu'une machine redèmarre au milieu d'une modification de table, vous devrez faire une vèrification de toutes les tables qui pourraient être affectèes : c'est une "table supposèement corrompue". Vous pouvez ajouter un test á `safe_mysqld` pour qu'il exècute `myisamchk`, afin de vèrifier toutes les tables qui ont ètè modifièes dans les 24 derniéres heures, si il reste un vieux fichier '`.pid`' (identifiant de processus) aprés un redèmarrage : le fichier '`.pid`' est crèè par le serveur `mysqld` lorsqu'il dèmarre, et il est supprimè lorsque le serveur s'arrête dans des conditions normales. La prèsence d'un fichier '`.pid`' au dèmarrage indique que le serveur s'est arrêtè anormalement.

Un test encore meilleur serait de vèrifier toutes les tables dont la date de modification est plus rècente que celle du fichier '`.pid`'.

Vous devriez aussi vèrifiez vos tables règuliérement durant les opèrations normales. Chez MySQL AB, nous utilisons une tâche en `cron` pour vèrifier toutes nos tables importantes au moins une fois par semaine, avec une ligne comme celle-ci dans le fichier '`crontab`' :

```
35 0 * * 0 /path/to/myisamchk --fast --silent /path/to/datadir/*/*.MYI
```

Cela nous affiche les informations sur les tables qui ont ètè corrompues, de façon á ce que nous puissions les examiner et les rèparer.

Comme nous n'avons jamais eu de table qui se soit corrompue inopinèment (des tables qui se corrompent pour d'autres raisons que des problémes matèriels) depuis quelques annèes (ce qui est vèridique), une fois par semaine est un bon rythme pour nous.

Nous recommandons que vous commenciez par exècuter la commande `myisamchk -s` chaque nuit, sur toutes les tables qui ont ètè modifièes dans les 24 derniéres heures, jusqu'á ce que vous preniez confiance en MySQL.

Normalement, vous n'avez pas á maintenir autant les tables MySQL. Si vous changez les tables avec un format de ligne dynamique (les tables avec des colonnes `VARCHAR`, `BLOB` ou

TEXT) ou que vous avez des tables avec de nombreuses lignes effacèes, vous pouvez envisager de faire des dèfragmentations du fichier, pour rècupèrer cet espace. Une fois par mois est un bon rythme.

Vous pouvez faire cela avec la commande SQL OPTIMIZE TABLE sur les tables en question, ou bien, si vous avez èteint le serveur mysqld, faites :

```
isamchk -r --silent --sort-index -O sort_buffer_size=16M */*.ISM
myisamchk -r --silent --sort-index  -O sort_buffer_size=16M */*.MYI
```

4.4.8 Obtenir des informations sur une table

Pour obtenir la description d'une table ou des statistiques á son sujet, utiliser les commandes affichèes ici. Nous allons expliquer certains de leurs dètails ultèrieurement.

- myisamchk -d nom_de_table Exècute myisamchk en "mode description" pour produire une description de votre table. Si vous dèmarrez le serveur MySQL en utilisant l'option --skip-external-locking, myisamchk va rapporter une erreur si la table est modifièe durant l'exècution de la commande. Cependant, comme myisamchk ne modifie pas les tables, durant le mode description, il n'y a pas de risque de perte de donnèes.
- myisamchk -d -v nom_de_table Pour produire plus d'informations durant l'exècution de myisamchk, ajoutez l'option -v pour indiquer qu'elle doit fonctionner en mode dètaillè.
- myisamchk -eis nom_de_table Affiche les informations les plus importantes pour une table. C'est une commande lente, car elle doit lire toute la table.
- myisamchk -eiv nom_de_table C'est l'èquivalent de -eis, mais qui vous indique ce qui se passe.

Exemple d'affichage rèsultant de myisamchk -d :

```
MyISAM file:     company.MYI
Record format:   Fixed length
Data records:    1403698  Deleted blocks:         0
Recordlength:    226

table description:
Key Start Len Index   Type
1   2     8   unique  double
2   15    10  multip. text packed stripped
3   219   8   multip. double
4   63    10  multip. text packed stripped
5   167   2   multip. unsigned short
6   177   4   multip. unsigned long
7   155   4   multip. text
8   138   4   multip. unsigned long
9   177   4   multip. unsigned long
    193   1           text
```

Exemple d'affichage rèsultant de myisamchk -d -v:

```
MyISAM file:         company
Record format:       Fixed length
```

```
File-version:        1
Creation time:       1999-10-30 12:12:51
Recover time:        1999-10-31 19:13:01
Status:              checked
Data records:           1403698  Deleted blocks:              0
Datafile parts:         1403698  Deleted data:                0
Datafilepointer (bytes):      3  Keyfile pointer (bytes):     3
Max datafile length: 3791650815  Max keyfile length: 4294967294
Recordlength:               226

table description:
Key Start Len Index   Type                     Rec/key     Root Blocksize
1   2     8   unique  double                         1 15845376      1024
2   15    10  multip. text packed stripped          2 25062400      1024
3   219   8   multip. double                        73 40907776      1024
4   63    10  multip. text packed stripped          5 48097280      1024
5   167   2   multip. unsigned short             4840 55200768      1024
6   177   4   multip. unsigned long              1346 65145856      1024
7   155   4   multip. text                       4995 75090944      1024
8   138   4   multip. unsigned long                87 85036032      1024
9   177   4   multip. unsigned long               178 96481280      1024
    193   1           text
```

Exemple d'affichage rèsultant de `myisamchk -eis`:

```
Checking MyISAM file: company
Key:  1:  Keyblocks used:  97%  Packed:    0%  Max levels:  4
Key:  2:  Keyblocks used:  98%  Packed:   50%  Max levels:  4
Key:  3:  Keyblocks used:  97%  Packed:    0%  Max levels:  4
Key:  4:  Keyblocks used:  99%  Packed:   60%  Max levels:  3
Key:  5:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
Key:  6:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
Key:  7:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
Key:  8:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
Key:  9:  Keyblocks used:  98%  Packed:    0%  Max levels:  4
Total:    Keyblocks used:  98%  Packed:   17%

Records:           1403698    M.recordlength:     226
Packed:              0%
Recordspace used:      100%   Empty space:          0%
Blocks/Record:   1.00
Record blocks:     1403698    Delete blocks:        0
Recorddata:      317235748    Deleted data:         0
Lost space:              0    Linkdata:             0

User time 1626.51, System time 232.36
Maximum resident set size 0, Integral resident set size 0
Non physical pagefaults 0, Physical pagefaults 627, Swaps 0
```

```
Blocks in 0 out 0, Messages in 0 out 0, Signals 0
Voluntary context switches 639, Involuntary context switches 28966
```

Exemple d'affichage rèsultant de `myisamchk -eiv`:

```
Checking MyISAM file: company
Data records: 1403698   Deleted blocks:       0
- check file-size
- check delete-chain
block_size 1024:
index  1:
index  2:
index  3:
index  4:
index  5:
index  6:
index  7:
index  8:
index  9:
No recordlinks
- check index reference
- check data record references index: 1
Key:  1:  Keyblocks used:  97%  Packed:    0%  Max levels:  4
- check data record references index: 2
Key:  2:  Keyblocks used:  98%  Packed:   50%  Max levels:  4
- check data record references index: 3
Key:  3:  Keyblocks used:  97%  Packed:    0%  Max levels:  4
- check data record references index: 4
Key:  4:  Keyblocks used:  99%  Packed:   60%  Max levels:  3
- check data record references index: 5
Key:  5:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
- check data record references index: 6
Key:  6:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
- check data record references index: 7
Key:  7:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
- check data record references index: 8
Key:  8:  Keyblocks used:  99%  Packed:    0%  Max levels:  3
- check data record references index: 9
Key:  9:  Keyblocks used:  98%  Packed:    0%  Max levels:  4
Total:    Keyblocks used:   9%  Packed:   17%

- check records and index references
[LOTS OF ROW NUMBERS DELETED]

Records:           1403698    M.recordlength:     226   Packed:              0%
Recordspace used:      100%   Empty space:          0%  Blocks/Record:   1.00
Record blocks:     1403698    Delete blocks:        0
Recorddata:      317235748    Deleted data:         0
```

```
Lost space:            0    Linkdata:            0

User time 1639.63, System time 251.61
Maximum resident set size 0, Integral resident set size 0
Non physical pagefaults 0, Physical pagefaults 10580, Swaps 0
Blocks in 4 out 0, Messages in 0 out 0, Signals 0
Voluntary context switches 10604, Involuntary context switches 122798
```

Voici les tailles des fichiers de données et d'index utilisès dans les tables prècèdentes :

```
-rw-rw-r--   1 monty    tcx     317235748 Jan 12 17:30 company.MYD
-rw-rw-r--   1 davida   tcx      96482304 Jan 12 18:35 company.MYM
```

Des dètails sur les types d'informations retournès par `myisamchk` sont listès ici. Le "keyfile" est le fichier d'index. "Record" et "row" sont synonymes de ligne :

- ISAM file Nom du fichier d'index ISAM.
- Isam-version Version du format ISAM. Actuellement, c'est toujours 2.
- Creation time Date de crèation du fichier de donnèes.
- Recover time Date de derniére reconstruction du fichier de donnèes ou d'index.
- Data records Combien de lignes sont stockèes dans la table.
- Deleted blocks Combien de blocs effacès occupent toujours de l'espace. Vous pouvez optimiser la table pour rècupèrer cet espace. Voir Section 4.4.6.10 [Optimisation], page 257.
- Data file: Parts Pour les tables au format de ligne dynamique, ceci indique combien de blocs de donnèes sont prèsents. Pour une table optimisèe sans lignes fragmentèes, la valeur doit être ègale á `Data records`.
- Deleted data Combien d'octets de donnèes effacèes et non rèutilisèes sont prèsents dans la table. Vous pouvez optimiser la table pour rècupèrer cet espace. Voir Section 4.4.6.10 [Optimisation], page 257.
- Data file pointer La taille du pointeur de fichier de donnèes, en octets. C'est gènèralement 2, 3, 4, ou 5 octets. La plupart des tables peuvent se gèrer avec 2 octets, mais ceci ne peut être contrôlè par MySQL actuellement. Pour les tables á format de ligne fixe, c'est une adresse de ligne. Pour les tables dynamiques, c'est une adresse d'octet.
- Keyfile pointer La taille du pointeur de fichier d'index, en octets. C'est gènèralement 1, 2 ou 3 octets. La plupart des tables supportent 2 octets, mais cela est calculè automatiquement par MySQL. C'est toujours une adresse de bloc.
- Max datafile length Taille maximale du fichier de donnèes ('`.MYD`' file), en octets.
- Max keyfile length Taille maximale du fichier d'index ('`.MYI`' file), en octets.
- Recordlength Taille occupèe par chaque ligne, en octets.
- Record format Le format utilisè pour stocker les lignes de la table. Les exemples ci-dessus utilisaient `Fixed length`. Les autres valeurs possibles sont `Compressed` et `Packed`.
- table description Une liste de toutes les clès de la table. Pour chaque clè, des informations de bas niveau sont prèsentèes :
  - Key Le numèro d'index.

  - Start Oú, dans la ligne, l'index dèbute.
  - Len Taille de cette partie d'index. Pour les nombres compactès, c'est toujours la taille maximale de la colonne. Pour les chaînes, c'est plus petit que la taille maximale de la colonne index, car vous pouvez indexer un prèfixe de la chaîne.
  - Index `unique` et `multip.` (multiple). Indique si une valeur peut exister plusieurs fois dans cet index.
  - Type De quel type de donnèes cet index est. C'est un type de donnèes ISAM avec les options `packed`, `stripped` ou `empty`.
  - Root Adresse du premier bloc d'index.
  - Blocksize La taille de chaque bloc d'index. Par dèfaut, c'est 1024, mais cette valeur peut être modifièe lors de la compilation.
  - Rec/key C'est une valeur statistique, utilisèe par l'optimisateur. Il indique combien de lignes sont disponibles par valeur de cette clè. Une clè unique aura toujours une valeur de 1. Cela peut être modifiè une fois que la table est chargèe (ou modifièe de façon majeure), avec la commande `myisamchk -a`. Si ce n'est pas mis á jour, une valeur par dèfaut de 30 est utilisèe.

- Dans le premier exemple ci-dessus, la neuviéme clè est une clè multi-partie, avec deux parties.
- Keyblocks used Quel pourcentage des blocs de clè est utilisè. Comme les tables utilisèes dans les exemples ont tout juste ètè rèorganisèes avec `myisamchk`, ces valeurs sont trés grandes (trés proches du maximum thèorique).
- Packed MySQL essaie de compacter les clès ayant un prèfixe commun. Cela ne peut être utilisè que pour les colonnes de type `CHAR/VARCHAR/DECIMAL`. Pour les longues chaînes comme des noms, cette technique va significativement rèduire l'espace utilisè. Dans le troisiéme exemple ci-dessus, la quatriéme clè fait 10 caractéres de long et a une rèduction de 60 % dans l'espace utilisè effectivement.
- Max levels La profondeur du B-tree. Les grandes tables avec de longues clès peuvent obtenir de grandes valeurs.
- Records Combien de lignes sont enregistrèes dans la table.
- M.recordlength La taille moyenne d'une ligne. Pour les tables avec un format de ligne statique, c'est la taille de chaque ligne.
- Packed MySQL efface les espaces á la fin des chaînes. `Packed` indique le pourcentage d'èconomie d'espace rèalisè.
- Recordspace used Quel est le pourcentage d'utilisation du fichier de donnèes.
- Empty space Quel est le pourcentage d'utilisation du fichier d'index.
- Blocks/Record Le nombre moyen de blocs par enregistrements (c'est á dire, de combien de liens une ligne fragmentèes est constituèe). C'est toujours 1.0 pour les tables á format de ligne statique. Cette valeur doit être aussi proche que possible de 1.0. Si elle grossit trop, vous pouvez rèorganiser la table avec `myisamchk`. Voir Section 4.4.6.10 [Optimisation], page 257.
- Recordblocks Combien de blocs sont utilisès. Pour les tables á format de ligne fixe, c'est le même nombre que le nombre de lignes.
- Deleteblocks Combien de blocs (liens) sont effacès.

- Recorddata Combien d'octets sont utilisès dans le fichier.
- Deleted data Combien d'octets dans le fichier de donnèes sont effacès (inutilisès).
- Lost space Si une ligne est modifièe, et rèduite en taille, de l'espace est perdu. Ce chiffre est la somme de ces espaces perdus, en octets.
- Linkdata Lorsque le format de ligne dynamique est utilisè, les fragments de lignes sont liès avec des pointeurs de (4 á 7 octets chacun). `Linkdata` est la somme du stockage utilisè par ces pointeurs.

Si une table a ètè compressèe avec `myisampack`, `myisamchk -d` affiche des informations supplèmentaires á propos de chaque colonne. Voir Section 4.7.4 [`myisampack`], page 297, pour un exemple de ces informations, et une description de leur signification.

4.5 Rèfèrence de langage d'administration de la base de donnèes

4.5.1 Syntaxe de `OPTIMIZE TABLE`

```
OPTIMIZE TABLE tbl_name[,tbl_name]...
```

`OPTIMIZE TABLE` doit être utilisèe si une grande partie de la base a ètè effacèe, ou si vous avez fait de nombreuses modificaitons dans une table á format de ligne dynamique (des tables qui ont des colonnes de type `VARCHAR`, `BLOB` et `TEXT`). Les lignes effacèes sont conservèes dans une liste, et les prochaines opèrations d'`INSERT` rèutilisent les vieilles positions de lignes. Vous pouvez vous servir de la commande `OPTIMIZE TABLE` pour rècupèrer l'espace utilisè et dèfragmenter le fichier de donnèes.

Pour le moment, `OPTIMIZE TABLE` fonctionne uniquement avec les tables de type `MyISAM` et `BDB`. Pour les tables `BDB`, `OPTIMIZE TABLE` est actuellement l'èquivalent de `ANALYZE TABLE`. Voir Section 4.5.2 [`ANALYZE TABLE`], page 264.

Vous pouvez vous arranger pour que `OPTIMIZE TABLE` fonctionne sur d'autres types de tables, en dèmarrant `mysqld` avec `--skip-new` ou `--safe-mode`, mais dans ce cas, `OPTIMIZE TABLE` est simplement l'èquivalent de `ALTER TABLE`.

`OPTIMIZE TABLE` fonctionne comme ceci :

- Si la table contient des lignes effacèes ou des lignes fragmentèes, la table est compactèe.
- Si les pages d'index ne sont pas trièes, `OPTIMIZE TABLE` les trie. Si les statistiques ne sont pas á jour (et que la table n'a pas pu effectuer de rèparation en triant l'index), elles sont mises á jour.

`OPTIMIZE TABLE` pour les tables `MyISAM` est l'èquivalent de la commande `myisamchk --quick --check-only-changed --sort-index --analyze` sur la table.

Notez que la table est verrouilèe durant la commande `OPTIMIZE TABLE`.

4.5.2 Syntaxe de `ANALYZE TABLE`

```
ANALYZE TABLE nom_de_table[,nom_de_table...]
```

Cette commande analyse et stocke la clè de distribution de la table. Durant l'analyse, la table est verrouilèe en lecture. Cette commande fonctionne avec les tables `MyISAM` et `BDB`.

C'est l'èquivalent de la commande en ligne `myisamchk -a`.

MySQL utilise les clès de distribution pour dècider dans quel ordre les tables doivent être rassemblèes lors des jointures qui ne s'effectuent pas sur une constante.

La commande retourne une table avec les colonnes suivantes :

Colonne	Valeur
Table	Nom de la table
Op	"analyze" (toujours)
Msg_type	Un des `status`, `error`, `info` ou `warning`.
Msg_text	Le message.

Vous pouvez vèrifier la clè de distribution stockèe avec la commande `SHOW INDEX`. Voir Section 4.5.6.1 [SHOW DATABASE INFO], page 268.

Si la table n'a pas changè depuis la derniére commande `ANALYZE TABLE`, elle ne sera pas analysèe á nouveau.

4.5.3 Syntaxe de `FLUSH`

```
FLUSH flush_option [,flush_option] ...
```

Vous devez utiliser la commande `FLUSH` si vous voulez effacer certains caches internes de MySQL. Pour exècuter `FLUSH`, vous devez avoir le droit `RELOAD`.

`flush_option` peut être l'une des suivantes :

Option	Description
`HOSTS`	Vide le cache des hôtes. Vous devez vider ce cache si certaines des adresses IP de vos clients changent, ou si vous obtenez des erreurs du type `Host ... is blocked`. Lorsque plus de `max_connect_errors` erreurs successives surviennent pour un hôte, lors des connexions au serveur MySQL, MySQL suppose qu'il y a un probléme, et interdit l'accés á l'hôte. Voir Section A.2.4 [Blocked host], page 688. Vous pouvez dèmarrer `mysqld` avec `-O max_connect_errors=999999999` pour èviter ce message.
`DES_KEY_FILE`	Recharge les clès DES depuis le fichier de stockage spècifiè par `--des-key-file` lors du dèmarrage du serveur.
`LOGS`	Ferme et rèouvre tous les fichiers de log. Si vous avez spècifiè un fichier de log de mise á jour, ou un fichier de log binaire sans extension, le numèro d'extension du fichier de log sera incrèmentè d'une unitè. Si vous avez utilisè une extension dans le nom du fichier, MySQL va fermer et rèouvrir le même fichier. Voir Section 4.9.3 [Update log], page 328. Ceci est la même chose que d'envoyer le signal `SIGHUP` au serveur `mysqld`.
`PRIVILEGES`	Recharge les priviléges des tables de droits dans la base `mysql`.

`QUERY CACHE`	Dèfragmente le cache des requêtes pour mieux en utiliser la mèmoire. Cette commande n'effacera aucune requête du cache, á la diffèrence de `RESET QUERY CACHE`.
`TABLES`	Ferme toutes les tables ouvertes, et force les tables utilisèes á se refermer. Cela vide aussi le cache de requêtes.
`[TABLE | TABLES] nom_de_table [,nom_de_table...]`	Vide du cache uniquement les tables nommèes.
`TABLES WITH READ LOCK`	Ferme toutes les tables ouvertes, et verrouille en lecture toute les tables et bases, jusqu'á ce que vous exècutiez une commande `UNLOCK TABLES`. C'est trés pratique pour gènèrer des sauvegardes, si vous avez un systéme de fichiers comme Veritas, qui peut prendre des photos du systéme.
`STATUS`	Remet la plupart des variables de statut á zèro. A n'utiliser que pour corriger une requête.
`USER_RESOURCES`	Remet toutes les ressources á zèro. Cela va autoriser de nouveau les utilisateurs qui ont ètè bloquès. Voir Section 4.3.6 [User resources], page 236.

Vous pouvez aussi accèder á toutes les commandes dècrites plus haut en les donnant en arguments á `mysqladmin` (exemple : `flush-hosts`, `flush-logs`, `reload`, ou encore `flush-tables`).

Reportez-vous aussi á la commande `RESET` avec la rèplication. Voir Section 4.5.4 [`RESET`], page 266.

4.5.4 Syntaxe de la commande `RESET`

```
RESET reset_option [,reset_option] ...
```

La commande `RESET` sert á remettre á zèro des donnèes. C'est aussi une version plus puissante de la commande `FLUSH`. Voir Section 4.5.3 [`FLUSH`], page 265.

Pour exècuter la commande `RESET`, vous devez avoir les droits `RELOAD`.

Option	**Description**
`MASTER`	Efface tous les logs binaires listès dans le fichier d'index, et l'index binlog est vidè. Dans les version antèrieures á la version 3.23.26, cette commande s'appelait `FLUSH MASTER` (Master)
`SLAVE`	Annule la position de rèplication de l'esclave dans les historiques du maître. Dans les version antèrieures á la version 3.23.26, cette commande s'appelait `FLUSH SLAVE`(Slave)
`QUERY CACHE`	Supprime tous les rèsultats de requêtes du cache de requête.

4.5.5 Syntaxe de `KILL`

```
KILL thread_id
```

Chaque connexion á `mysqld` utilise un thread unique. Vous pouvez voir les threads en cours d'exècution en utilisant la commande `SHOW PROCESSLIST` et en terminer un avec la commande `KILL thread_id`.

Si vous avez le droit `PROCESS`, vous pouvez voir tous les threads. Si vous avez le droit `SUPER`, vous pouvez terminer tout les threads. Sinon, vous ne pouvez terminer que vos propres threads.

Vous pouvez aussi utiliser les commandes `mysqladmin processlist` et `mysqladmin kill` pour examiner et terminer les threads.

Quand vous exècutez un `KILL`, un thread spècifique est crèe pour ce thread.

Dans la plupart des cas, la terminaison du thread pourra prendre un certain temps vu que le thread de terminaison est invoquè á intervalles spècifiques.

- Pour les boucles de `SELECT`, `ORDER BY` et `GROUP BY`, le thread de terminaison est vèrifiè aprés avoir lu un enregistrement. S'il est activè la requête est abandonnèe.
- Lors d'un `ALTER TABLE` le thread de terminaison est vèrifiè avant la lecture de chacune des colonnes de la table d'origine. S'il est activè, la commande est abandonnèe et la table temporaire effacèe.
- Lors d'un `UPDATE TABLE` ou d'un `DELETE TABLE`, le thread de terminaison est vèrifiè aprés chaque lecture de bloc et chaque mise á jour ou suppression de ligne. S'il est activè, la requête est abandonnèe. Notez que si vous utilisez les transactions, les modifications ne seront pas perdues !
- `GET_LOCK()` stoppera avec `NULL`.
- Un thread `INSERT DELAYED` videra rapidement toutes les lignes en mèmoire et se terminera.
- Si le thread est dans le gestionnaire des verrous de tables (ètat : `Locked`), le verrou sur la table sera vite enlevè.
- Si le thread est en attente de libèration d'espace disque lors d'un appel á `write`, l'opèration est avortèe avec un message d'erreur indiquant que le disque est plein.

4.5.6 Syntaxe de `SHOW`

```
   SHOW DATABASES [LIKE wild]
ou SHOW [OPEN] TABLES [FROM nom_base] [LIKE wild]
ou SHOW [FULL] COLUMNS FROM nom_de_table [FROM nom_base] [LIKE wild]
ou SHOW INDEX FROM nom_de_table [FROM nom_base]
ou SHOW TABLE STATUS [FROM nom_base] [LIKE wild]
ou SHOW STATUS [LIKE wild]
ou SHOW VARIABLES [LIKE wild]
ou SHOW LOGS
ou SHOW [FULL] PROCESSLIST
ou SHOW GRANTS FOR user
ou SHOW CREATE TABLE table_name
ou SHOW MASTER STATUS
ou SHOW MASTER LOGS
ou SHOW SLAVE STATUS
```

SHOW fournit des informations sur les bases de donnèes, les colonnes, les tables, ou l'ètat du serveur. Si la clause LIKE wild est utilisèe, la chaîne wild peut être une chaîne qui utilise les caractéres SQL joker '%' et '_'.

4.5.6.1 Obtenir des informations sur les bases, tables, colonnes et index

Vous pouvez utiliser la syntaxe alternative db_name.tbl_name á la syntaxe tbl_name FROM db_name. Ces deux commandes sont èquivalentes :

```
mysql> SHOW INDEX FROM mytable FROM mydb;
mysql> SHOW INDEX FROM mydb.mytable;
```

SHOW DATABASES liste les bases que le serveur MySQL hèberge. Vous pouvez aussi obtenir cette liste en utilisant la commande mysqlshow. Depuis la version 4.0.2, vous ne verrez que les bases pour lesquelles vous avez des droits, á moins que vous n'ayez le droit global de SHOW DATABASES.

SHOW TABLES liste les tables d'une base de donnèes. Vous pouvez aussi accèder á cette liste avec la commande mysqlshow db_name.

Note : si un utilisateur n'a aucun droit sur une table, la table ne sera pas affichèe dans le rèsultat de SHOW TABLES et mysqlshow db_name.

SHOW OPEN TABLES liste les tables qui sont actuellement ouvertes dans le cache de table. Voir Section 5.4.7 [Table cache], page 388. Le champs Comment indique combien de fois la table a ètè mise en cache (cached) et utilisèe (in_use).

SHOW COLUMNS liste les colonnes d'une table donnèes. Si vous spècifiez l'option FULL, vous obtiendrez aussi les droits dont vous disposez sur chaque colonne. Si les types de colonnes sont diffèrents de ce á quoi vous vous attendiez aprés votre requête CREATE TABLE, notez que MySQL change parfois les types de colonnes. Voir Section 6.5.3.1 [Silent column changes], page 511.

La requête DESCRIBE fournit des informations similaires á SHOW COLUMNS. Voir Section 6.6.2 [DESCRIBE], page 518.

SHOW FIELDS est un synonyme de SHOW COLUMNS, et SHOW KEYS est un synonyme de SHOW INDEX. Vous pouvez aussi lister les colonnes d'une table ou ses index avec les commandes en ligne mysqlshow db_name tbl_name et mysqlshow -k db_name tbl_name.

SHOW INDEX retourne le dètails sur les index, dans un format qui ressemble vaguement á l'affichage obtenu par la commande SQLStatistics de ODBC. Les colonnes suivantes sont disponibles :

Colonne	Signification
Table	Nom de la table.
Non_unique	0 si l'index ne peut pas contenir de doublons.
Key_name	Nom de l'index.
Seq_in_index	Numèro de sèquence de la colonne dans l'index, commençant á 1.
Column_name	Nom de la colonne.
Collation	Comment la colonne est placèe dans l'index. Avec MySQL, les valeurs peuvent être 'A' (Ascending) ou NULL (Non triè).

`Cardinality`	Nombre de valeurs uniques dans l'index. Ceci est mis á jour en exècutant la commande `isamchk -a`.
`Sub_part`	Numbre de caractéres indexès si la colonne est partiellement indexèe. `NULL` siginifie que la clèe entiére est indexèe.
`Null`	Contient 'YES' si la colonne peut contenir la valeur `NULL`.
`Index_type`	La mèthode d'indexation utilisèe.
`Comment`	Diffèrentes remarques. Actuellement, il indique si l'index est `FULLTEXT` ou non (MySQL < 4.0.2).

Notez que la cardinalitè (`Cardinality`) est comptèe grâce á des statistiques stockèes sous forme d'entier. Ce n'est peut être pas trés prècis pour les petites tables.

Les colonnes `Null` et `Index_type` ont ètè ajoutèes en MySQL 4.0.2.

4.5.6.2 SHOW TABLE STATUS

```
SHOW TABLE STATUS [FROM db_name] [LIKE wild]
```

`SHOW TABLE STATUS` (nouveau en version 3.23) fonctionne comme `SHOW STATUS`, mais fournit des informations sur les tables. Vous pouvez aussi obtenir ces informations en utilisant la commande en ligne `mysqlshow --status db_name`. Les donnèes suivantes sont retournèes :

Colonne	**Signification**
`Name`	Nom de la table.
`Type`	Type de table. Voir Chapitre 7 [Table types], page 531.
`Row_format`	Le format de stockage de ligne (Fixed, Dynamic ou Compressed).
`Rows`	Nombre de lignes.
`Avg_row_length`	Taille moyenne d'une ligne.
`Data_length`	Taille du fichier de donnèes.
`Max_data_length`	Taille maximale du fichier de donnèes.
`Index_length`	Taille du fichier d'index.
`Data_free`	Nombre d'octets allouès mais non utilisès.
`Auto_increment`	Prochaine valeur d'autoincrement.
`Create_time`	Date de crèation de la table.
`Update_time`	Date de derniére modification de la table.
`Check_time`	Date de dernier entretien de la table.
`Create_options`	Options supplèmentaires utilisèes avec `CREATE TABLE`.
`Comment`	Le commentaire utilisè lors de la crèation de la table (ou des informations sur pourquoi MySQL n'a pu accèder aux informations de la table).

Les tables `InnoDB` indiqueront l'espace disque libre dans le commentaire de table.

4.5.6.3 Syntaxe de SHOW STATUS

`SHOW STATUS` affiche des informations sur le statut du serveur (comme par exemple, `mysqladmin extended-status`). L'affichage ressemble á ce qui est affichè ci-dessous, mais les valeurs diffèreront sûrement de votre propre serveur.

```
+--------------------------+------------+
| Variable_name            | Value      |
+--------------------------+------------+
| Aborted_clients          | 0          |
| Aborted_connects         | 0          |
| Bytes_received           | 155372598  |
| Bytes_sent               | 1176560426 |
| Connections              | 30023      |
| Created_tmp_disk_tables  | 0          |
| Created_tmp_tables       | 8340       |
| Created_tmp_files        | 60         |
| Delayed_insert_threads   | 0          |
| Delayed_writes           | 0          |
| Delayed_errors           | 0          |
| Flush_commands           | 1          |
| Handler_delete           | 462604     |
| Handler_read_first       | 105881     |
| Handler_read_key         | 27820558   |
| Handler_read_next        | 390681754  |
| Handler_read_prev        | 6022500    |
| Handler_read_rnd         | 30546748   |
| Handler_read_rnd_next    | 246216530  |
| Handler_update           | 16945404   |
| Handler_write            | 60356676   |
| Key_blocks_used          | 14955      |
| Key_read_requests        | 96854827   |
| Key_reads                | 162040     |
| Key_write_requests       | 7589728    |
| Key_writes               | 3813196    |
| Max_used_connections     | 0          |
| Not_flushed_key_blocks   | 0          |
| Not_flushed_delayed_rows | 0          |
| Open_tables              | 1          |
| Open_files               | 2          |
| Open_streams             | 0          |
| Opened_tables            | 44600      |
| Questions                | 2026873    |
| Select_full_join         | 0          |
| Select_full_range_join   | 0          |
| Select_range             | 99646      |
| Select_range_check       | 0          |
| Select_scan              | 30802      |
| Slave_running            | OFF        |
| Slave_open_temp_tables   | 0          |
| Slow_launch_threads      | 0          |
| Slow_queries             | 0          |
| Sort_merge_passes        | 30         |
```

```
| Sort_range               | 500        |
| Sort_rows                | 30296250   |
| Sort_scan                | 4650       |
| Table_locks_immediate    | 1920382    |
| Table_locks_waited       | 0          |
| Threads_cached           | 0          |
| Threads_created          | 30022      |
| Threads_connected        | 1          |
| Threads_running          | 1          |
| Uptime                   | 80380      |
+--------------------------+------------+
```

Les variables de statut listèes ci-dessus ont la signification suivante :

Variable	Signification
`Aborted_clients`	Nombre de connexions annulèes parce que le client est mort sans se dèconnecter correctement. Voir Section A.2.9 [Communication errors], page 690.
`Aborted_connects`	Nombre de tentatives de connexions au serveur MySQL qui ont èchouèes. Voir Section A.2.9 [Communication errors], page 690.
`Bytes_received`	Nombre d'octets reçu de tous les clients.
`Bytes_sent`	Nombre d'octets envoyès á tous les clients.
`Com_xxx`	Nombre d'exècution de chaque commande.
`Connections`	Nombre de tentatives de connexions au serveur MySQL.
`Created_tmp_disk_tables`	Nombre de tables temporaires implicites crèèes sur le disque lors d'exècutions de commandes.
`Created_tmp_tables`	Nombre de tables temporaires implicites crèèes en mèmoire lors d'exècutions de commandes.
`Created_tmp_files`	Combien de fichiers temporaires `mysqld` a crèè.
`Delayed_insert_threads`	Nombre de gestionnaires d'insertion retardèes sont en cours d'utilisation.
`Delayed_writes`	Nombre de lignes ècrites avec `INSERT DELAYED`.
`Delayed_errors`	Nombre de lignes ècrites avec `INSERT DELAYED` pour lesquelles des erreurs sont survenues (probablement une erreur de doublons (`duplicate key`)).
`Flush_commands`	Nombre de commandes `FLUSH`.
`Handler_commit`	Nombre de commandes internes `COMMIT`.
`Handler_delete`	Nombre de fois qu'une ligne a ètè effacèes dans une table.
`Handler_read_first`	Nombre de fois que la premiére ligne a ètè lue dans un index. Si ce chiffre est haut, c'est que le serveur fait de nombreuses recherches par analyse compléte de la table, par exemple `SELECT col1 FROM foo`, en supposant que col1 est indexè.
`Handler_read_key`	Nombre de requête de lecture de ligne basèes sur une clè. Si ce chiffre est grand, c'est une bonne indication de l'indexation correcte de vos tables.
`Handler_read_next`	Nombre de requête de lecture de la ligne suivante en ordre. Cela sera augmentè si vous listez une colonne avec une contrainte d'intervalle. Cette valeur sera aussi incrèmentèe si vous effectuez un scan d'index.

`Handler_read_prev`	Nombre de requête de lecture de la clè prècèdente, dans l'ordre. C'est souvent utilisè pour optimiser les clauses `ORDER BY ... DESC`.
`Handler_read_rnd`	Nombre de lecture d'une ligne basèe sur une position fixe. Ce chiffre sera grand si vous effectuez de nombreuses requêtes qui rèclament le tri du rèsultat.
`Handler_read_rnd_next`	Nombre de requêtes de lecture de la prochaine ligne dans le fichier de donnèes. Ce chiffre sera grand si vous faîtes de nombreux scans de tables. Gènèralement, cela indique que vos requêtes ne sont pas ècrites pour profiter des index que vous avez mis en place.
`Handler_rollback`	Nombre de commandes internes `ROLLBACK`.
`Handler_update`	Nombre de requête de modification d'une ligne dans une table.
`Handler_write`	Nombre de requête pour insèrer une ligne dans une table.
`Key_blocks_used`	The Nombre de blocs utilisès dans un cache de clè.
`Key_read_requests`	The Nombre de requêtes de lecture d'un bloc de clè dans le cache.
`Key_reads`	Nombre de lecture physique d'un bloc de clè sur le disque.
`Key_write_requests`	Nombre de requêtes d'ècriture d'un bloc de clè dans le cache.
`Key_writes`	Nombre d'ècriture physiques de bloc de clè sur le disque.
`Max_used_connections`	Nombre maximum de connexions utilisèes simultanèment.
`Not_flushed_key_blocks`	Nombre de blocs de clès dans le cache de clès, qui ont ètè modifièes, mais pas encore ècrites sur le disque.
`Not_flushed_delayed_rows`	Nombre de lignes en attente d'ècriture dans les listes `INSERT DELAY`.
`Open_tables`	Nombre de tables ouvertes.
`Open_files`	Nombre de fichiers ouverts.
`Open_streams`	Nombre de flôts ouverts (utilisès gènèralement pour les logs).
`Opened_tables`	Nombre de tables qui ont ètè ouvertes.
`Rpl_status`	Statut de la rèplication sans erreur (rèservè pour utilisation ultèrieure).
`Select_full_join`	Nombre de jointures sans clè (si cette variable vaut 0, vous devriez vèrifier soigneusement les index de vos tables).
`Select_full_range_join`	Nombre de jointures oú une recherche d'intervalle a ètè utilisèe.
`Select_range`	Nombre de jointures oú une rercherche d'intervalle a ètè utilisèe sur la premiére table. (Ce n'est gènèralement pas important, même si cette valeur est importante).
`Select_scan`	Nombre de jointures oú la premiére table a ètè totalement scannèe.
`Select_range_check`	Nombre de jointures sans clè, oú l'utilisation de clè a ètè vèrifièe aprés chaque ligne (si cette variable vaut 0, vous devriez vèrifier soigneusement les index de vos tables).
`Questions`	Nombre de requêtes envoyèes au serveur.
`Slave_open_temp_tables`	Nombre de tables temporaires actuellement utilisèe par le thread esclave.
`Slave_running`	Cette variable vaut `ON` si ce serveur est un esclave connectè au maître.
`Slow_launch_threads`	Nombre de threads qui ont pris plus de `slow_launch_time` secondes pour être crèès.

`Slow_queries`	Nombre de requêtes qui ont pris plus de `long_query_time` pour s'exècuter. Voir Section 4.9.5 [Slow query log], page 331.
`Sort_merge_passes`	Nombre de passes que l'algorithme de tri a du faire. Si cette valeur est grande, vous devriez vèrifier la taille de `sort_buffer`.
`Sort_range`	Nombre de tris qui ont ètè fait sur des intervalles.
`Sort_rows`	Nombre de lignes triées.
`Sort_scan`	Nombre de tris qui ont ètè fait en scannant la table.
`ssl_xxx`	Variables utilisèes par SSL; Rèservèe pour utilisation ultèrieure.
`Table_locks_immediate`	Nombre de fois que la table a reçu immèdiatement un verrou. Disponible depuis 3.23.33.
`Table_locks_waited`	Nombre de fois qu'une table n'a pu recevoir de verrou immèdiatement, et qu'il a fallu attendre. Si ce chiffre est haut, vous avez des problémes de performance, et vous devriez optimiser vos requêtes, couper vos tables en deux, ou utiliser la rèplication. Disponible depuis la version 3.23.33.
`Threads_cached`	Nombre de threads dans le cache de thread.
`Threads_connected`	Nombre de connexions actuellement ouvertes.
`Threads_created`	Nombre de threads crèès pour gèrer les connexions.
`Threads_running`	Nombre de threads qui ne dorment pas.
`Uptime`	Durèe de vie du serveur, en secondes depuis le redèmarrage.

Quelques commentaires sur les variables ci-dessus :

- Si `Opened_tables` est grand, alors votre `table_cache` est probablement trop petit.
- Si `Key_reads` est grand, alors `key_buffer_size` est probablement trop petit. Le taux d'utilisation du cache peut être calculè avec le ratio `Key_reads/Key_read_requests`.
- Si `Handler_read_rnd` est grand, vous avez probablement trop de requêtes qui analysent toute la table, ou vous avez des jointures qui n'exploitent pas d'index.
- Si `Threads_created` est grand, vous devriez augmenter la valeur de la variable `thread_cache_size`. Le taux d'utilisation du cache peut être calculè avec `Threads_created/Connections`.
- Si `Created_tmp_disk_tables` est grand, vous devriez augmenter la valeur de la variable `tmp_table_size` pour obtenir des tables temporaires en mèmoire, et non plus sur le disque.

4.5.6.4 Syntaxe de `SHOW VARIABLES`

```
SHOW [GLOBAL | SESSION] VARIABLES [LIKE wild]
```

`SHOW VARIABLES` affiche les valeurs des variables systémes de MySQL. Vous pouvez aussi obtenir ces informations avec la commande `mysqladmin variables`. Si les valeurs par dèfaut ne vous conviennent pas, vous pouvez modifier la plupart de ces variables, en ligne de commande, lorsque `mysqld` est lancè. Voir Section 4.1.1 [Command-line options], page 192.

Les options `GLOBAL` et `SESSION` sont nouvelles depuis MySQL 4.0.3. Avec `GLOBAL` vous allez lister les variables qui sont utilisèes pour les nouvelles connexions á MySQL. Avec `SESSION`, vous allez lister les valeurs qui ont actuellement cours pour la connexion courante. Si vous n'utilisez ni l'une, ni l'autre des options, `SESSION` sera utilisèe.

Vous pouvez changer la plupart des options avec la commande `SET`. Voir Section 5.5.6 [SET], page 396.

Le rèsultat de cette commande ressemble á la table ci-dessous, même si les nombres et formats différent un peu :

```
+---------------------------------+------------------------------+
| Variable_name                   | Value                        |
+---------------------------------+------------------------------|
| back_log                        | 50                           |
| basedir                         | /usr/local/mysql             |
| bdb_cache_size                  | 8388572                      |
| bdb_log_buffer_size             | 32768                        |
| bdb_home                        | /usr/local/mysql             |
| bdb_max_lock                    | 10000                        |
| bdb_logdir                      |                              |
| bdb_shared_data                 | OFF                          |
| bdb_tmpdir                      | /tmp/                        |
| bdb_version                     | Sleepycat Software: ...   |
| binlog_cache_size               | 32768                        |
| bulk_insert_buffer_size         | 8388608                      |
| character_set                   | latin1                       |
| character_sets                  | latin1 big5 czech euc_kr  |
| concurrent_insert               | ON                           |
| connect_timeout                 | 5                            |
| convert_character_set           |                              |
| datadir                         | /usr/local/mysql/data/       |
| delay_key_write                 | ON                           |
| delayed_insert_limit            | 100                          |
| delayed_insert_timeout          | 300                          |
| delayed_queue_size              | 1000                         |
| flush                           | OFF                          |
| flush_time                      | 0                            |
| ft_min_word_len                 | 4                            |
| ft_max_word_len                 | 254                          |
| ft_max_word_len_for_sort        | 20                           |
| ft_boolean_syntax               | + -><()~*:""&|               |
| have_bdb                        | YES                          |
| have_innodb                     | YES                          |
| have_isam                       | YES                          |
| have_raid                       | NO                           |
| have_symlink                    | DISABLED                     |
| have_openssl                    | YES                          |
| have_query_cache                | YES                          |
| init_file                       |                              |
| innodb_additional_mem_pool_size | 1048576                      |
| innodb_buffer_pool_size         | 8388608                      |
| innodb_data_file_path           | ibdata1:10M:autoextend       |
```

```
| innodb_data_home_dir            |                              |
| innodb_file_io_threads          | 4                            |
| innodb_force_recovery           | 0                            |
| innodb_thread_concurrency       | 8                            |
| innodb_flush_log_at_trx_commit  | 0                            |
| innodb_fast_shutdown            | ON                           |
| innodb_flush_method             |                              |
| innodb_lock_wait_timeout        | 50                           |
| innodb_log_arch_dir             |                              |
| innodb_log_archive              | OFF                          |
| innodb_log_buffer_size          | 1048576                      |
| innodb_log_file_size            | 5242880                      |
| innodb_log_files_in_group       | 2                            |
| innodb_log_group_home_dir       | ./                           |
| innodb_mirrored_log_groups      | 1                            |
| interactive_timeout             | 28800                        |
| join_buffer_size                | 131072                       |
| key_buffer_size                 | 16773120                     |
| language                        | /usr/local/mysql/share/...   |
| large_files_support             | ON                           |
| local_infile                    | ON                           |
| locked_in_memory                | OFF                          |
| log                             | OFF                          |
| log_update                      | OFF                          |
| log_bin                         | OFF                          |
| log_slave_updates               | OFF                          |
| log_slow_queries                | OFF                          |
| log_warnings                    | OFF                          |
| long_query_time                 | 10                           |
| low_priority_updates            | OFF                          |
| lower_case_table_names          | OFF                          |
| max_allowed_packet              | 1047552                      |
| max_binlog_cache_size           | 4294967295                   |
| max_binlog_size                 | 1073741824                   |
| max_connections                 | 100                          |
| max_connect_errors              | 10                           |
| max_delayed_threads             | 20                           |
| max_heap_table_size             | 16777216                     |
| max_join_size                   | 4294967295                   |
| max_sort_length                 | 1024                         |
| max_user_connections            | 0                            |
| max_tmp_tables                  | 32                           |
| max_write_lock_count            | 4294967295                   |
| myisam_max_extra_sort_file_size | 268435456                    |
| myisam_max_sort_file_size       | 2147483647                   |
| myisam_recover_options          | force                        |
| myisam_sort_buffer_size         | 8388608                      |
```

```
| net_buffer_length               | 16384                          |
| net_read_timeout                | 30                             |
| net_retry_count                 | 10                             |
| net_write_timeout               | 60                             |
| open_files_limit                | 0                              |
| pid_file                        | /usr/local/mysql/name.pid      |
| port                            | 3306                           |
| protocol_version                | 10                             |
| read_buffer_size                | 131072                         |
| read_rnd_buffer_size            | 262144                         |
| rpl_recovery_rank               | 0                              |
| query_cache_limit               | 1048576                        |
| query_cache_size                | 0                              |
| query_cache_type                | ON                             |
| safe_show_database              | OFF                            |
| server_id                       | 0                              |
| slave_net_timeout               | 3600                           |
| skip_external_locking           | ON                             |
| skip_networking                 | OFF                            |
| skip_show_database              | OFF                            |
| slow_launch_time                | 2                              |
| socket                          | /tmp/mysql.sock                |
| sort_buffer_size                | 2097116                        |
| sql_mode                        | 0                              |
| table_cache                     | 64                             |
| table_type                      | MYISAM                         |
| thread_cache_size               | 3                              |
| thread_stack                    | 131072                         |
| tx_isolation                    | READ-COMMITTED                 |
| timezone                        | EEST                           |
| tmp_table_size                  | 33554432                       |
| tmpdir                          | /tmp/                          |
| version                         | 4.0.4-beta                     |
| wait_timeout                    | 28800                          |
+---------------------------------+--------------------------------+
```

Tous les options sont dècrites ici. Les tailles de buffer, les longueurs et piles sont donnèes en octets. Vous pouvez spècifier ces valeurs avec le suffixe '`K`' ou '`M`' pour indiquer respectivement kilo-octets ou mèga-octets. Par `16M` représente 16 mègaoctets. Le suffixe peut être en majuscule ou en minuscule. `16M` et `16m` sont èquivalents :

- `ansi_mode`. Vaut `ON` si `mysqld` a ètè dèmarrè en mode `--ansi`. Voir Section 1.7.2 [ANSI mode], page 33.
- `back_log` Le nombre de connexions sortantes que MySQL peut supporter. Cette valeur entre en jeu lorsque le thread principal MySQL reçoit de **trés** nombreuses requêtes de connexions en trés peu de temps. MySQL prend un peu de temps (même si c'est trés peu de temps), pour vèrifier la connexion et dèmarrer un nouveau thread. La valeur de `back_log` indique combien de requête seront mises en attente durant ce temps. Vous

devrez augmenter ce nombre si vous voulez mettre en attente plus de requêtes durant une courte pèriode de temps.

En d'autres termes, cette valeur est la taille de la queue d'attente pour les connexions TCP/IP entrantes. Votre systéme d'exploitation a ses propres limites pour ce type de queue. La page du manuel Unix `listen(2)` doit contenir plus de dètails. Vèrifiez la documentation de votre OS pour connaître la valeur maximale de votre systéme. Si vous donne une valeur á `back_log` qui est plus grande que celle que votre systéme supporte, cela restera sans effet.

- `basedir` La valeur de l'option `--basedir`.
- `bdb_cache_size` Le buffer qui est allouè pour mettre en cache des lignes et des index pour les tables `BDB`. Si vous n'utilisez pas la tables `BDB`, vous devriez dèmarrer `mysqld` avec l'option `--skip-bdb` pour ne pas gaspiller de mèmoire.
- `bdb_log_buffer_size` Le buffer qui est allouè pour mettre en cache des lignes et des index pour les tables `BDB`. Si vous n'utilisez pas la tables `BDB`, vous devriez dèmarrer `mysqld` avec l'option `--skip-bdb` pour ne pas gaspiller de mèmoire.
- `bdb_home` La valeur de l'option `--bdb-home`.
- `bdb_max_lock` Le nombre maximum de verrous (par dèfaut 10 000) que vous pouvez activer simultanèment dans une table `BDB`. Vous devriez augmenter cette valeur si vous obtenez des erreurs du type `bdb: Lock table is out of available locks` ou `Got error 12 from ...` lorsque vous avez de longues transactions ou que `mysqld` doit examiner de nombreuses lignes pour calculer la requête.
- `bdb_logdir` La valeur de l'option `--bdb-logdir`.
- `bdb_shared_data` Vaut `ON` si vous utilisez l'option `--bdb-shared-data`.
- `bdb_tmpdir` La valeur de l'option `--bdb-tmpdir`.
- `binlog_cache_size`. La taille du cache qui contient les requêtes SQL destinèes au log binaire, durant une transaction. Si vous utilisez souvent de longues transactions multi-requêtes, vous devriez augmenter cette valeur pour amèliorer les performances. Voir Section 6.7.1 [COMMIT], page 518.
- `bulk_insert_buffer_size` (ètait `myisam_bulk_insert_tree_size`) `MyISAM` utilise une cache hièrarchisè pour les insertions de masses (c'est á dire `INSERT ... SELECT`, `INSERT ... VALUES (...), (...), ...`, et `LOAD DATA INFILE`). Cette variable limite la taille du cache en octets, par threads. Utiliser la valeur de 0 va dèsactiver cette optimisation. **Note** : ce cache est uniquement utilisè lorsque vous ajoutez des donnèes dans une table non-vide. Par dèfaut, cette option vaut 8 Mo.
- `character_set` Le jeu de caractéres par dèfaut.
- `character_sets` Les jeux de caractéres supportès.
- `concurrent_inserts` Si cette option vaut `ON`, MySQL va vous permettre de rèaliser des commandes `INSERT` sur les tables `MyISAM` en même temps que d'autres commandes `SELECT` seront exècutées. Vous pouvez dèsactiver cette option en dèmarrant `mysqld` avec l'option `--safe` or `--skip-new`.
- `connect_timeout` Le nombre de secondes d'attente d'un paquet de connexion avant de conclure avec une erreur `Bad handshake`.
- `datadir` La valeur de l'option `--datadir`.

- `delay_key_write` Les options pour les tables `MyISAM`. Elles peuvent prendre l'une des valeurs suivantes :

`OFF`	Toutes les commandes CREATE TABLE ... DELAYED_KEY_WRITES sont ignorès.
`ON`	(par dèfaut) MySQL va honorer l'option `DELAY_KEY_WRITE` de `CREATE TABLE`.
`ALL`	Toutes les nouvelles tables ouvertes sont traitèes comme si elles ètaient crèèes avec l'option `DELAY_KEY_WRITE`.

  Si `DELAY_KEY_WRITE` est activè, cela signifie que le buffer de clè des tables ayant cette option ne seront pas ècrit sur le disque dés la fin de la modification de la table, mais attendrons que la table soit ècrite. Cela accèlére notablement les ècritures des modifications, mais il faut penser á ajouter une vèrification automatique des tables au dèmarrage avec `myisamchk --fast --force`.

- `delayed_insert_limit` Aprés avoir insèrè `delayed_insert_limit` lignes, le gestionnaire de `INSERT DELAYED` va vèrifiez si il n'y a pas de commande `SELECT` en attente. Si c'est le cas, il va autoriser ces commandes avant de continuer.

- `delayed_insert_timeout` Combien de temps le thread `INSERT DELAYED` doit attendre les commandes `INSERT` avant de s'achever.

- `delayed_queue_size` Quelle taille de file (en lignes) doit être allouèe pour gèrer les commandes `INSERT DELAYED`. Si la file se remplit, tous les clients qui èmettent des commandes `INSERT DELAYED` devront attendre un peu de place avant de pouvoir continuer.

- `flush` Cette option vaut `ON` si vous avez dèmarrè MySQL avec l'option `--flush`.

- `flush_time` Si cette option a une valeur non nulle, toutes les `flush_time` secondes, toutes les tables seront fermèes (pour libèrez des ressources et synchroniser les index sur le disque). Nous ne recommandons cette option que sur les systémes Windows 9x/Me, ou les systémes qui ont trés peu de ressource.

- `ft_min_word_len` La taille minimale d'un mot inclus dans un index `FULLTEXT`. **Note : les index `FULLTEXT` doivent être reconstruit aprés avoir modifiè cette variable.** (Cette option est nouvelle en MySQL 4.0.)

- `ft_max_word_len` La taille maximale d'un mot inclus dans un index `FULLTEXT`. **Note : les index `FULLTEXT` doivent être reconstruit aprés avoir modifiè cette variable.** (Cette option est nouvelle en MySQL 4.0.)

- `ft_max_word_len_for_sort` La taille maximale d'un mot dans un index `FULLTEXT`, pour qu'il soit inclus dans la mèthode de recrèation rapide de l'index, avec la commande `REPAIR`, `CREATE INDEX`, ou `ALTER TABLE`. Les mots longs sont insèrès par une mèthode plus lente. La régle idèale est la suivante : avec `ft_max_word_len_for_sort` qui croit, **MySQL** va crèer des fichiers temporaires plus grand, et donc, ralentir le processus, á cause des accés disques, et cela va mettre moins de clès dans un bloc de tri (encore une fois, limitant la vitesse). Lorsque `ft_max_word_len_for_sort` est plus petit, **MySQL** va insèrer un grand nombre de mot avec la mèthode lente, mais les mots courts seront insèrès trés vite.

- `ft_boolean_syntax` Liste des opèrateurs supportès par `MATCH ... AGAINST(... IN BOOLEAN MODE)`. Voir Section 6.8 [Fulltext Search], page 522.

- `have_innodb YES` si `mysqld` supporte les tables InnoDB. `DISABLED` si `--skip-innodb` est utilisèe.
- `have_bdb YES` si `mysqld` supporte les tables Berkeley. `DISABLED` si `--skip-bdb` est utilisèe.
- `have_raid YES` si `mysqld` supporte l'option `RAID`.
- `have_openssl YES` si `mysqld` supporte le chiffrement SSL entre le client et le serveur.
- `init_file` Le nom du fichier spècifiè avec l'option `--init-file` lorsque vous dèmarrez le serveur. C'est un fichier qui contient les requêtes SQL que vous voulez voir exècutèes dés le dèmarrage.
- `interactive_timeout` Le nombre de secondes durant lequel le serveur attend une activitè de la part de la connexion avant de la fermèe. Un client interactif est un client qui utilise l'option `CLIENT_INTERACTIVE` avec `mysql_real_connect()`. Voir aussi `wait_timeout`.
- `join_buffer_size` La taille du buffer qui est utilisèe pour les jointures complétes (les jointures qui n'utilisent pas d'index). Ce buffer est allouè une fois pour chaque jointure entre deux tables. Augmentez cette valeur si vous voulez obtenir des jointures plus rapides, lorsque l'ajout d'index n'est pas possible. Normalement, le mieux est d'ajouter de bons index.
- `key_buffer_size` Les blocs d'index sont mis en buffer et partagès par tous les threads. `key_buffer_size` est la taille du buffer utilisè.

  Augmentez cette valeur pour obtenir une meilleure gestion des index (pour les lectures et ècritures multiples), autant que vous le pouvez : 64Mo sur une machine de 256Mo est une valeur rèpandue. Toutefois, si vous utilisez une valeur trop grande (par exemple, plus de 50% de votre mèmoire totale), votre systéme risque de commencer á utiliser sa mèmoire swap, et devenir trés lent. N'oubliez pas que MySQL ne met pas en cache les donnèes lues, et il faut laisser le systéme d'exploitation respirer.

  Vous pouvez vèrifier les performances du buffer de clès avec la commande `show status` et en examinant les variables `Key_read_requests`, `Key_reads`, `Key_write_requests` et `Key_writes`. Le ratio `Key_reads/Key_read_request` doit être normalement < 0.01. Le ratio `Key_write/Key_write_requests` est gènèralement proche de 1 si vous utilisez principalement des modifications et effacement, mais il peut être bien plus petit si vous faites des modifications qui affectent de nombreuses lignes en même temps, ou si vous utilisez l'option `DELAY_KEY_WRITE`. Voir Section 4.5.6 [`SHOW`], page 267.

  Pour obtenir encore plus de vitesse lors de l'ècriture de plusieurs lignes en même temps, utilisez `LOCK TABLES`. Voir Section 6.7.2 [`LOCK TABLES`], page 519.
- `language` La langue utilisèe pour les message d'erreurs.
- `large_file_support` Si `mysqld` a ètè compilè avec le support des grands fichiers.
- `locked_in_memory` Si `mysqld` a ètè verrouillè en mèmoire avec `--memlock`
- `log` Si le log de toutes les requêtes est dèsactivè.
- `log_update` Si le log de modification est activè.
- `log_bin` Si le log binaire est activè.
- `log_slave_updates` Si les modifications des esclaves doivent être enregistrèes.
- `long_query_time` Si une requête prend plus de

- `long_query_time` secondes, le compteur de requêtes lentes `Slow_queries` sera incrèmentè. Si vous utilisez l'option `--log-slow-queries`, ces requêtes seront enregistrèes dans un historique de requêtes lentes. Cette durèe est mesurèe en temps rèel, et non pas en temps processus, ce qui fait que les requêtes qui seraient juste sous la limite avec un systéme lègérement chargè, pourrait être au dessus avec le même systéme, mais chargè. Voir Section 4.9.5 [Slow query log], page 331.
- `lower_case_table_names` Si cette option vaut 1, les noms de tables sont stockèes en minuscules sur le disque, et les comparaisons de nom de tables seront insensibles á la casse. Depuis la version 4.0.2, cette option s'applique aussi aux noms de bases. Voir Section 6.1.3 [Name case sensitivity], page 408.
- `max_allowed_packet` La taille maximale d'un paquet. Le buffer de message est initialisè avec `net_buffer_length` octets, mais peut grandir jusqu'á `max_allowed_packet` octets lorsque nècessaire. Cette valeur est par dèfaut petit, pour intercepter les gros paquets, probablement erronès. Vous devez augmenter cette valeur si vous utilisez de grandes colonnes `BLOB`. Cette valeur doit être aussi grande que le plus grand `BLOB` que vous utiliserez. Le protocole limite actuellement `max_allowed_packet` á 16Mo en MySQL 3.23 et 1Go en MySQL 4.0.
- `max_binlog_cache_size` Si une transaction multi-requête requiert plus que cette quantitè de mèmoire, vous obtiendrez une erreur `"Multi-statement transaction required more than 'max_binlog_cache_size' bytes of storage"`.
- `max_binlog_size` Disponible depuis la version 3.23.33. Si vous ècrivez dans le log binaire (de rèplication) et que cela dèpasse la taille de `max_binlog_size`, une erreur sera indiquèe. Vous ne pouvez pas donner á `max_binlog_size` une valeur infèrieure á 1024 octets, ou plus grande que 1 Go.
- `max_connections` Le nombre maximal de clients simultanès acceptè. En augmentant cette valeur, vous augmentez le nombre de pointeur de fichier que requiert `mysqld`. Voyez plus bas les commentaires sur les pointeurs de fichiers. Voir Section A.2.5 [Too many connections], page 689.
- `max_connect_errors` Si il y a plus que `max_connect_errors` connexion interrompues depuis un même hôte, cet hôte sera bloquè dans ses prochaines tentatives de connexions. Vous pouvez dèbloquer un hôte avec la commande `FLUSH HOSTS`.
- `max_delayed_threads` Ne pas lancer plus que `max_delayed_threads` threads pour gèrer les insertions `INSERT DELAYED`. Si vous essayez d'insèrer des donnèes dans une nouvelle table alors que tous les gestionnaires `INSERT DELAYED` sont utilisès, la ligne sera insèrèe comme si l'option `DELAYED` n'avait pas ètè spècifièe.
- `max_heap_table_size` Ne pas autoriser la crèation de tables de type HEAP plus grande que `max_heap_table_size`.
- `max_join_size` Les jointures qui liront probablement plus de `max_join_size` lignes, retourneront une erreur. Utilisez cette valeur si vos utilisateurs font des jointures avec de mauvaises clauses `WHERE`, qui prennent trop de temps, et retournent des millions de lignes.
- `max_sort_length` Le nombre d'octets á utiliser lors du tri des colonnes de type `BLOB` et `TEXT`. Seuls les `max_sort_length` octets de chaque valeur seront utilisès pour le tri. Le reste est ignorè.

- `max_user_connections` Le nombre maximum de connexions actives pour un utilisateur particulier (0 = pas de limite).
- `max_tmp_tables` (Cette option ne fait actuellement rien du tout). Le nombre maximum de tables temporaires qu'un client peut garder ouvertes en même temps.
- `max_write_lock_count` Aprés `max_write_lock_count` pose de verrou en ècriture, autorise quelques verrous en lecture.
- `myisam_recover_options` La valeur de l'option `--myisam-recover`.
- `myisam_sort_buffer_size` Le buffer qui est allouès lors du tri d'index avec la commande `REPAIR` ou lors de la crèation d'index avec `CREATE INDEX` ou `ALTER TABLE`.
- `myisam_max_extra_sort_file_size`. Si un fichier temporaire utilisè pour la crèation rapide d'index devient plus grand que le cache de clè spècifiè ici, la mèthode utilisant le cache de clè sera utilisè. C'est principalement utilisè pour forcer les index longs de grandes tables á utiliser une mèthode plus lente pour ècrire l'index. **Notez** que ce paramétre est spècifiè en mègaoctets avant la version 4.0.3 et en octets depuis cette version.
- `myisam_max_sort_file_size` La taille maximale du fichier temporaire que MySQL est autorisè á utiliser durant la recrèation des fichiers d'index (avec `REPAIR`, `ALTER TABLE` ou `LOAD DATA INFILE`). Si la taille du fichier dèpasse `myisam_max_sort_file_size`, l'index sera crèè avec un cache de clè (plus lent). **Notez** que ce paramétre est spècifiè en mègaoctets avant la version 4.0.3 et en octets depuis.
- `net_buffer_length` Le buffer de communication est remis á zèro entre deux requêtes. Cela ne devrait pas être modifiè, mais si vous avez trés peu de mèmoire, vous pouvez le remettre á la taille prèsumèe de la requête (c'est á dire, la taille de requête envoyè par le client. Si la requête dèpasse cette taille, le buffer est automatiquement agrandi jusqu'á `max_allowed_packet` octets).
- `net_read_timeout` Nombre de secondes d'attente des derniéres donnèes, avant d'annuler la lecture. Notez que lorsque nous n'attendons pas de donnèes d'une connexion, le dèlai d'expiration est donnè par `write_timeout`. Voir aussi `slave_net_timeout`.
- `net_retry_count` Si une lecture sur une port de communication est interrompu, `net_retry_count` tentatives sont faites avant d'abandonner. Cette valeur doit être particuliérement grande pour `FreeBSD` car les interruptions internes sont envoyès á tous les threads.
- `net_write_timeout` Nombre de secondes d'attente pour qu'un bloc soit envoyè á une connexion, avant d'annuler l'ècriture.
- `open_files_limit` Si `open_files_limit` ne vaut pas 0, alors `mysqld` va utiliser cette valeur pour rèserver des pointeurs de fichiers á utiliser avec `setrlimit()`. Si cette valeur est 0, alors `mysqld` va rèserver `max_connections*5` ou `max_connections + table_cache*2` (le plus grand des deux) pointeurs de fichiers. Vous devriez augmenter cette valeur si `mysqld` vous donne des erreurs du type `'Too many open files'`.
- `pid_file` La valeur de l'option `--pid-file`.
- `port` La valeur de l'option `--port`.
- `protocol_version` La version du protocole utilisè par le serveur MySQL.

- `read_buffer_size` (was `record_buffer`) Chaque thread qui doit faire une recherche sèquentielle alloue une buffer de cette taille pour chaque table qu'il scanne. Si vous faîtes de nombreuses recherches sèquentielles, vous devriez augmenter cette valeur.
- `record_rnd_buffer_size` Lors de la lecture de lignes qui sont placèes dans un ordre triè, les lignes sont lues via ce buffer pour èviter les recherches sur le disque. `record_rnd_buffer_size` peut amèliorer grandement les performances de la clause `ORDER BY` si cette valeur est grande. Comme c'est une variable qui est spècifique pour les threads, il ne faut pas trop l'augmenter, mais simplement la modifier lorsque de grandes requêtes sont exècutèes.
- `query_cache_limit` Ne met pas en cache les rèsultats qui sont plus grands que `query_cache_limit`. Par dèfaut, 1 Mo.
- `query_cache_size` La mèmoire allouèe pour stocker les rèsultats des vieilles requêtes. Si `query_cache_size` vaut 0, le cache de requête est dèsactivè (par dèfaut).
- `query_cache_type` `query_cache_type` peut prendre les valeurs numèriques suivantes :

Valeur	Alias	Commentaire
0	`OFF`	Ne met pas en cache les rèsultats.
1	`ON`,	Met en cache tous les rèsultats exceptès les requêtes `SELECT SQL_NO_CACHE ...`.
2	`DEMAND`	Met en cache uniquement les requêtes `SELECT SQL_CACHE ...`.

- `safe_show_database` Ne montre pas les bases pour lesquelles un utilisateur n'a pas des droits de bases ou de tables. Cela peut amèliorer considèrablement la sècuritè si vous craignez de voir les utilisateurs dècouvrir ce que les autres ont mis en place. Voir aussi `skip_show_database`.
- `server_id` La valeur de l'option `--server-id`.
- `skip_locking` `skip_locking` vaut OFF si `mysqld` utilise le verrouillage externe.
- `skip_networking` `skip_networking` vaut ON si seules les connexions locales (via socket) sont autorisèes.
- `skip_show_database` `skip_show_database` empêche les utilisateurs d'exècuter des commandes `SHOW DATABASES` si ils n'ont pas les droits de `PROCESS`. Cela peut amèliorer la sècuritè si vous craignez de voir les utilisateurs dècouvrir ce que les autres ont mis en place. Voir aussi `safe_show_database`.
- `slave_net_timeout` Nombre de secondes d'attente de donnèes en lecture ou ècriture sur une connexion maître / esclave avant d'annuler.
- `slow_launch_time` Si la crèation du thread prend plus de `slow_launch_time` secondes, le compteur de threads lents `Slow_launch_threads` sera incrèmentè.
- `socket` La socket Unix utilisè par le serveur.
- `sort_buffer` Chaque thread qui doit faire un tri alloue un buffer de cette taille. Augmentez cette taille pour accèlèrer les clauses `ORDER BY` ou `GROUP BY`. Voir Section A.4.4 [Temporary files], page 701.
- `table_cache` Le nombre de tables ouvertes pour tous les threads rèunis. En augmentant cette valeur, vous augmentez le nombre de pointeurs de fichiers que `mysqld` utilise. Vous pouvez vèrifier si vous avez besoin de plus de cache de tables en ètudiant la valeur de la variable `Opened_tables`. Voir Section 4.5.6 [SHOW], page 267. Si cette variable

est grande, est que vous ne faites pas souvent de commande `FLUSH TABLES` (qui force les tables á se recharger), vous devrez alors augmenter cette valeur.

Pour plus d'informations sur le cache de table, voyez Section 5.4.7 [Table cache], page 388.

- `table_type` Le type de table par dèfaut.
- `thread_cache_size` Combien de threads nous allons conserver en cache pour rèutilisation. Lorsqu'un client se dèconnecte, les threads du client sont mis en cache s'il n'y en a pas dèjá `thread_cache_size` de conservè. Tous les nouveaux threads sont d'abord prèlevè dans le cache, et uniquement lorsque le cache est vide, un nouveau thread est crèè. Cette variable peut vous permettre d'amèliorer les performances si vous avez de nombreuses connexions. Normalement, `thread_cache_size` ne donne pas d'amèlioration notable si vous avez une bonne implèmentation des threads. En examinant la diffèrence entre les variables de statut `Connections` et `Threads_created` vous pouvez voir comment votre systéme de cache de threads est efficace.
- `thread_concurrency` Sous Solaris, `mysqld` va appeler `thr_setconcurrency()` avec cette valeur. `thr_setconcurrency()` permet á l'application de donner au systéme de threads une indication sur le nombre de threads qui seront exècutès en même temps.
- `thread_stack` La taille de la pile pour chaque thread. De nombreuses limites dètectèes par `crash-me` sont dèpendantes de cette valeur. La valeur par dèfaut est suffisamment grande pour des opèrations normales. Voir Section 5.1.4 [MySQL Benchmarks], page 360.
- `timezone` Le fuseau horaire du serveur.
- `tmp_table_size` Si une table temporaire en mèmoire excéde cette taille, MySQL va automatiquement la convertir en une table `MyISAM` sur le disque. Augmentez la valeur de `tmp_table_size` si vous faites un usage intensif de la clause `GROUP BY` et que vous avez beaucoup de mèmoire.
- `tmpdir` Le dossier utilisè pour les fichiers temporaires et les tables temporaires.
- `version` Le numèro de version du serveur.
- `wait_timeout` Le nombre de secondes d'attente du serveur sur une connexion non interactive avant de la refermer.

  Lors du dèmarrage du thread, `SESSION.WAIT_TIMEOUT` est initialisè avec `GLOBAL.WAIT_TIMEOUT` ou `GLOBAL.INTERACTIVE_TIMEOUT`, suivant le type de client (tel que dèfini par l'option de connexion `CLIENT_INTERACTIVE`). Voir aussi `interactive_timeout`.

La section du manuel qui est consacrèe á l'optimisation de MySQL contient des informations sur l'utilisation des variables ci-dessus. Voir Section 5.5.2 [Server parameters], page 390.

4.5.6.5 Syntaxe de `SHOW LOGS`

La commande `SHOW LOGS` affiche les informations d'ètat de vos fichiers de logs. Actuellement, elle n'affiche que les informations pour les fichiers de log des tables Berkeley DB.

- `File` affiche le chemin complet jusqu'au fichier de log.
- `Type` affiche le type de fichier de log (`BDB` pour les tables de types Berkeley DB)
- `Status` affiche le status du fichier de log (`FREE` si le fichier peut être supprimè, ou `IN USE` si le fichier est utilisè par une transaction en cours)

4.5.6.6 Syntaxe de SHOW PROCESSLIST

`SHOW [FULL] PROCESSLIST` affiche la liste de processus qui sont en cours d'exècution. Vous pouvez aussi obtenir ces informations avec la commande en ligne `mysqladmin processlist`. Si vous avez les droits de `SUPER`, vous pourrez aussi voir les autres threads. Sinon, vous ne pourrez voir que les votre. Voir Section 4.5.5 [KILL], page 266. Si vous n'utilisez pas l'option `FULL`, seuls les 100 premiers caractéres de chaque requête seront affichès.

Cette commande est trés pratique si vous obtenez trop d'erreurs 'too many connections' et que vous voulez savoir ce qui se passe. MySQL rèserve une connexion supplèmentaire pour un client ayant les droits de `SUPER`, de façon á ce qu'il y ait toujours la possibilitè de se connecter et de vèrifier le systéme (en supposant que vous ne donnez pas ce droit á tous vos utilisateurs).

Certains ètats sont souvent disponible dans le rèsultat de `mysqladmin processlist`

- `Checking table` Le thread fait une vèrification (automatique) de la table.
- `Closing tables` Le thread est en train d'ècrire les donnèes modifièes sur le disque, et il va fermer les tables. Cela doit être une opèration trés rapide. Si ce n'est pas le cas, vous devriez vèrifier si vous n'avez pas un disque plein, ou que le disque est sous haute charge.
- `Connect Out` Connexion d'un esclave sur le maître.
- `Copying to tmp table on disk` Le rèsultat temporaire ètait plus grand que `tmp_table_size` et le thread passe d'une table en mèmoire á une table sur disque.
- `Creating tmp table` Le thread est en train de crèer une table temporaire pour contenir le rèsultat d'une requête.
- `deleting from main table` Lors de l'exècution de la premiére partie d'une requête d'effacement multi-table, et que MySQL n'a commencè á effacer que dans la premiére table.
- `deleting from reference tables` Lors de l'exècution de la deuxiéme partie d'une requête d'effacement multi-table, et que MySQL a commencè á effacer dans les autres tables.
- `Flushing tables` Le thread exècute la commande `FLUSH TABLES` et il attend que tous les threads ferme leur tables.
- `Killed` Quelqu'un a envoyè une commande `KILL` et le thread s'annuler la prochaine fois qu'il vèrifie l'option de kill. Cette option est vèrifièe dans chaque boucle majeure de MySQL, mais dans certains cas, il peut lui prendre un court instant avant de s'arrêter. Si le thread est verrouillè par un autre thread, l'arrêt va prendre effet aussitôt que l'autre thread léve son verrou.
- `Sending data` Le thread traite des lignes pour une commande `SELECT` et il envoie les donnèes au client.
- `Sorting for group` Le thread est en train de faire un tri pour satisfaire une clause `GROUP BY`.
- `Sorting for order` Le thread est en train de faire un tri pour satisfaire une clause `ORDER BY`.
- `Opening tables` Cela signifie simplement que le thread essaie d'ouvrir une table. Ce doit être une opèration trés rapide, á moins que quelque chose ne retarde l'ouverture.

Par exemple, une commande `ALTER TABLE` ou `LOCK TABLE` peut empêcher l'ouverture de table, jusqu'á l'achévement de la commande.

- `Removing duplicates` La requête utilisait `SELECT DISTINCT` de telle maniére que MySQL ne pouvait pas optimiser les lignes distinctes au dèbut du traitement. A cause de cela, MySQL doit effectuer une opèration de plus pour supprimer toutes les lignes en doubles, avant d'envoyer les lignes au client.
- `Reopen table` Le thread a reçu un verrou pour une table, mais a notè aprés l'avoir reçu que la structure de la table a changè. Il a libèrè le verrou, fermè la table, et maintenant il essaie de la rèouvrir.
- `Repair by sorting` Le thread rèpare la table en utilisant la mèthode de tri pour crèer l'index.
- `Repair with keycache` Le thread rèpare la table en utilisant la mèthode de crèation des clès á partir du cache de clè. C'est bien plus lent que `la rèparation par tri`.
- `Searching rows for update` Le thread effectue une premiére phase pour trouver toutes les lignes qui satisfont les critéres avant de les modifier. Cela doit être fait si `UPDATE` modifie l'index qui sera utilisè pour trouver les lignes.
- `Sleeping` Le thread attend que le client envoie une nouvelle commande.
- `System lock` Le thread attend le verrou externe pour la table. Si vous n'utilisez pas de serveurs MySQL multiples qui exploitent les mêmes tables, vous pouvez dèsactiver les verrous systémes avec l'option `--skip-external-locking`.
- `Upgrading lock` Le gestionnaire de `INSERT DELAYED` essaie d'obtenir un verrou pour insèrer des lignes.
- `Updating` Le thread recherche des lignes pour les modifier.
- `User Lock` Le thread attend un `GET_LOCK()`.
- `Waiting for tables` Le thread a reçu l'annonce que la structure de table a ètè modifièe, et il doit rèouvrir la table pour obtenir une nouvelle structure. Pour être capable de rèouvrir la table, il doit attendre que les autres threads aient fermè la table en question.

  Cette annonce survient lorsqu'un autre autre thread a ètè utilisè avec la commande `FLUSH TABLES` ou une des commandes suivantes, appliquèes á la table en question : `FLUSH TABLES table_name`, `ALTER TABLE`, `RENAME TABLE`, `REPAIR TABLE`, `ANALYZE TABLE` ou `OPTIMIZE TABLE`.
- `waiting for handler insert` Le gestionnaire de `INSERT DELAYED` a traitè toutes insertions, et en attend de nouvelles.

La plupart des ètats sont des opèrations trés rapides. Si le thread s'attarde dans un de ces ètats pour plusieurs secondes, il doit y avoir un probléme qui mèrite d'être ètudiè.

Il existe encore d'autres ètats qui ne sont pas mentionnè ci-dessus, mais la majoritè sont utilisès pour trouver des bogues dans `mysqld`.

4.5.6.7 SHOW GRANTS

`SHOW GRANTS FOR user` affiche la commande nècessaire pour donner les même droits qu'un utilisateur existant.

```
mysql> SHOW GRANTS FOR root@localhost;
+---------------------------------------------------------------------+
| Grants for root@localhost                                           |
+---------------------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION |
+---------------------------------------------------------------------+
```

4.5.6.8 Syntaxe de SHOW CREATE TABLE

Affiche la commande CREATE TABLE nècessaire pour crèer une table donnèe.

```
mysql> SHOW CREATE TABLE t\G
*************************** 1. row ***************************
       Table: t
Create Table: CREATE TABLE t (
  id int(11) default NULL auto_increment,
  s char(60) default NULL,
  PRIMARY KEY (id)
) TYPE=MyISAM
```

SHOW CREATE TABLE va protèger le nom de la table et des colonnes selon l'option SQL_QUOTE_SHOW_CREATE. Section 5.5.6 [SET SQL_QUOTE_SHOW_CREATE], page 396.

4.6 Localisation de MySQL et utilisation internationale

4.6.1 Le jeu de caractéres utilisè pour les donnèes et le stockage

Par dèfaut, MySQL utilise le jeu de caractéres ISO-8859-1 (Latin1) avec tri en accord au Suèdois/Finnois. C'est le jeu de caractére le mieux adaptè pour les USA et l'Europe de l'ouest.

Tous les binaires standards MySQL sont compilès avec `--with-extra-charsets=complex`. Cela ajoutera du code á tous les programmes standards pour qu'ils puissent gèrer `latin1` et tous les jeux de caractéres multi-octets compris dans le binaire. Les autres jeux de caractéres seront chargès á partir d'un fichier de dèfinition de jeu si besoin.

Le jeu de caractéres dètermine quels caractéres sont autorisès dans les noms et comment s'effectuent les tris dans les clauses ORDER BY et GROUP BY de la commande SELECT.

Vous pouvez changer le jeu de caractéres avec l'option de dèmarrage du serveur `--default-character-set`. Les jeux de caractéres disponibles dèpendent des options `--with-charset=charset` et `--with-extra-charsets= list-of-charset | complex | all` de `configure`, et des fichiers de configuration de jeux de caractéres situès dans '`SHAREDIR/charsets/Index`'. Voir Section 2.3.3 [configure options], page 88.

Si vous changez le jeu de caractéres lors de l'utilisation de MySQL (ce qui pourra aussi changer l'ordre de tri), vous devez exècuter `myisamchk -r -q --set-character-set=charset` sur toutes les tables. Sinon, vos index pourront ne pas être ordonnès correctement.

Lorsqu'un client se connecte á un serveur MySQL, le serveur envoie le jeu de caractéres utilisè par dèfaut au client. Le client changera de jeu de caractéres pour cette connexion.

Vous devez utiliser `mysql_real_escape_string()` pour protèger les chaînes pour une requête SQL. `mysql_real_escape_string()` est identique á l'ancienne fonction `mysql_escape_string()`, exceptè qu'elle prend le gestionnaire de connexion `MYSQL` en tant que premier paramétre.

Si le client est compilè avec d'autres chemins que ceux oú le serveur est installè et que la personne qui a configurè MySQL n'a pas inclut tous les jeux de caractéres dans le binaire MySQL, vous devez indiquer au client oú il peut trouver les jeux de caractéres additionnels dont il aura besoin si le serveur utilise un autre jeu de caractéres que le client.

On peut le spècifier en plaçant dans un fichier d'options MySQL :

```
[client]
character-sets-dir=/usr/local/mysql/share/mysql/charsets
```

oú le chemin pointe vers le rèpertoire oú les jeux de caractéres dynamiques de MySQL sont stockès.

On peut forcer le client á utiliser un jeu de caractéres spècifique en prècisant :

```
[client]
default-character-set=character-set-name
```

mais on n'en a normalement jamais besoin.

b

4.6.1.1 Jeu de caractéres allemand

Pour obtenir l'ordre de tri Allemand, vous devez dèmarrer `mysqld` avec `--default-character-set=latin1_de`. Cela vous donnera les caractèristiques diffèrentes.

Lors du tri et de la comparaison des chaînes, les remplacements suivants sont faits dans la chaîne avant d'effectuer la comparaison :

```
ä  ->  ae
ö  ->  oe
ü  ->  ue
ß  ->  ss
```

Tous les caractéres accentuès sont convertis en leur majuscule non-accentuèe. Toutes les lettres sont transformèes en majuscules.

Lors de la comparaison des chaînes de caractéres avec `LIKE` le conversion un `->` deux caractéres n'est pas effectuèe. Toutes les lettres sont transformèes en majuscules. Les accents sont supprimès de toutes les lettres, á l'exception de : `Ü`, `ü`, `Ö`, `ö`, `Ä` et `ä`.

4.6.2 Langue des messages d'erreurs

`mysqld` peut èmettre des messages d'erreurs dans les langues suivantes : Tchéque, Danois, Nèerlandais, Anglais (par dèfaut), Estonien, Français, Allemand, Grec, Hongrois, Italien, Japonais, Corèen, Norvègien, Norwegian-ny, Polonais, Portugais, Roumain, Russe, Slovaque, Espagnol et Suèdois.

Pour dèmarrer `mysqld` avec une langue particuliére, utilisez soit l'option `--language=lang`, soit `-L lang`. Par exemple :

```
shell> mysqld --language=french
```

ou :

```
shell> mysqld --language=/usr/local/share/french
```

Notez que tout les noms de langue sont spècifiès en miniscule.

Les fichiers de langue sont situès (par dèfaut) dans '*mysql_base_dir*/share/*LANGUAGE*/'.

Pour modifier le fichier de messages d'erreurs, vous devez èditer le fichier '`errmsg.txt`' et exècuter la commande suivante pour gènèrer le fichier '`errmsg.sys`' :

```
shell> comp_err errmsg.txt errmsg.sys
```

Si vous changez de version de MySQL, pensez á modifier le nouveau fichier '`errmsg.txt`'.

4.6.3 Ajouter un nouveau jeu de caractéres

Pour ajouter un autre jeu de caractéres á MySQL, utilisez la procèdure suivante.

Dècidez s'il s'agit d'un jeu simple ou complexe. Si le jeu de caractéres n'a pas besoin d'utiliser des routines d'assemblage de chaînes spèciales pour le tri et n'a pas besoin du support des jeux de caractéres multi-octets, il est simple. S'il a besoin de l'une de ces deux fonctionnalitès, il est complexe.

Par exemple, `latin1` et `danish` sont des jeux de caractéres simples tandis que `big5` et `czech` sont complexes.

Dans la section suivante, nous supposons que vous nommez votre jeu de caractéres `MONJEU`.

Pour un jeu de caractéres simple, effectuez ce qui suit :

1. Ajoutez MONJEU á la fin du fichier '`sql/share/charsets/Index`' Assignez-lui un nombre unique.
2. Crèez le fichier '`sql/share/charsets/MONJEU.conf`'. (Vous pouvez vous inspirer de '`sql/share/charsets/latin1.conf`'.)

   La syntaxe pour le fichier est trés simple :

   - Les commentaires commencent avec le caractére '#' et se terminent á la fin de la ligne.
   - Les mots sont sèparès par un nombre changeant d'espaces blancs.
   - Lors de la dèfinition d'un jeu de caractéres, chaque mot doit être un nombre au format hexadècimal.
   - Le tableau `ctype` prends les 257 premiers mots. Les tableaux `to_lower[]`, `to_upper[]` et `sort_order[]` prennent chacun 256 mots aprés cela.

   Voir Section 4.6.4 [Character arrays], page 289.
3. Ajoutez le nom du jeu de caractéres aux listes `CHARSETS_AVAILABLE` et `COMPILED_CHARSETS` dans `configure.in`.
4. Reconfigurez, recompilez et testez.

Pour un jeu de caractéres complexe faites ce qui suit :

1. Crèez le fichier '`strings/ctype-MONJEU.c`' dans la distribution des sources MySQL.

2. Ajoutez MONJEU á la fin du fichier '`sql/share/charsets/Index`'. Assignez-lui un nombre unique.
3. Regardez un des fichiers '`ctype-*.c`' existant pour voir ce qui doit être dèfini, par exemple, '`strings/ctype-big5.c`'. Notez que les tableaux dans votre fichier doivent avoir des noms tels que `ctype_MONJEU`, `to_lower_MONJEU`, etc. Cela correspond aux tableaux dans les jeux de caractéres simples. Voir Section 4.6.4 [Character arrays], page 289. Pour un jeu de caractére complexe
4. Au dèbut du fichier, placez un commentaire spècial comme celui-ci :

```
/*
 * This comment is parsed by configure to create ctype.c,
 * so don't change it unless you know what you are doing.
 *
 * .configure. number_MONJEU=MYNUMBER
 * .configure. strxfrm_multiply_MONJEU=N
 * .configure. mbmaxlen_MONJEU=N
 */
```

   Le programme `configure` utilise ce commentaire pour inclure automatiquement le jeu de caractéres dans la librairie MySQL.

   Les lignes strxfrm_multiply et mbmaxlen seront expliquèes dans les sections suivantes. Ne les incluez que si vous avez besoin des fonctions d'assemblage des chaînes ou des fonctions de jeu de caractéres multi-octets, respectivement.
5. Vous devez alors crèer les fonctions suivantes :
   - `my_strncoll_MONJEU()`
   - `my_strcoll_MONJEU()`
   - `my_strxfrm_MONJEU()`
   - `my_like_range_MONJEU()`

   Voir Section 4.6.5 [String collating], page 290.
6. Ajoutez le nom du jeu de caractéres aux listes `CHARSETS_AVAILABLE` et `COMPILED_CHARSETS` dans `configure.in`.
7. Reconfigurez, recompilez et testez.

Le fichier '`sql/share/charsets/README`' fournit plus d'instructions.

Si vous voulez qu'un jeu de caractéres soit ajoutè dans la distribution MySQL, envoyez un patch á `internals@lists.mysql.com`.

4.6.4 Le tableau de dèfinition des caractéres

`to_lower[]` et `to_upper[]` sont de simples tableaux qui contiennent les caractéres minuscules et majuscules correspondant á chaque membre du jeu de caractére. Par exemple :

```
to_lower['A'] doit contenir 'a'
to_upper['a'] doit contenir 'A'
```

`sort_order[]` est une carte indiquant comment les caractéres doivent être ordonnès pour les comparaisons et les tris. Pour beaucoup de jeux de caractéres, c'est la même chose

que `to_upper[]` (ce qui signifie que le tri sera insensible á la casse). MySQL triera les caractéres en se basant sur la valeur de `sort_order[caractére]`. Pour des régles de tri plus compliquèes, voyez la discussion suivante sur l'assemblage des chaînes. Voir Section 4.6.5 [String collating], page 290.

`ctype[]` est un tableau de valeurs de bit, avec un èlèment par caractére. (Notez que `to_lower[]`, `to_upper[]`, et `sort_order[]` sont indexès par la valeur du caractére, mais que `ctype[]` est indexè par la valeur du caractére + 1. C'est une vielle habitude pour pouvoir gèrer `EOF`.)

Vous pouvez trouver les dèfinitions de bitmask suivantes dans '`m_ctype.h`' :

```
#define _U      01      /* Majuscule */
#define _L      02      /* Minuscule */
#define _N      04      /* Numèrique (nombre) */
#define _S      010     /* Caractére d'espacement */
#define _P      020     /* Ponctuation */
#define _C      040     /* Caractére de contrôle */
#define _B      0100    /* Blanc */
#define _X      0200    /* nombre hexadecimal */
```

L'entrèe `ctype[]` de chaque caractére doit être l'union des valeurs de masque de bits qui dècrivent le caractére. Par exemple, `'A'` est un caractére majuscule (`_U`) autant qu'une valeur hexadècimale (`_X`), et donc `ctype['A'+1]` doit contenir la valeur :

```
_U + _X = 01 + 0200 = 0201
```

4.6.5 Support d'assemblage des chaînes

Si les régles de tri de votre langue sont trop complexes pour être gèrèes par le simple tableau `sort_order[]`, vous devez utiliser les fonctions d'assemblage de chaînes.

Jusqu'á prèsent, la meilleure documentation traitant de ce sujet est prèsente dans les jeux de caractéres implèmentès eux-mêmes. Regardez les jeux de caractéres `big5`, `czech`, `gbk`, `sjis`, et `tis160` pour des exemples.

Vous devez spècifier la valeur de `strxfrm_multiply_MYSET=N` dans le commentaire spècial au dèbut du fichier. `N` doit être le rationnel maximal vers lequel la chaîne pourra croître durant `my_strxfrm_MYSET` (cela doit être un entier positif).

4.6.6 Support des caractéres multi-octets

Si vous voulez ajouter le support de jeu de caractéres incluant des caractéres multi-octets, vous devez utiliser les fonctions de caractéres multi-octets.

Jusqu'á prèsent, la meilleure documentation traitant de ce sujet est prèsente dans les jeux de caractéres implèmentès eux-mêmes. Regardez les jeux de caractéres `euc_kr`, `gb2312`, `gbk`, `sjis`, et `ujis` pour des exemples. Ils sont implèmentès dans les fichiers '`ctype-'charset'.c`' dans le dossier '`strings`'.

Vous devez spècifier la valeur de `mbmaxlen_MYSET=N` dans le commentaire spècial en haut du fichier source. `N` doit être la taille en octet du caractére le plus large dans le jeu.

4.6.7 Problémes avec les jeux de caractéres

Si vous essayez d'utiliser un jeu de caractéres qui n'est pas compilè dans votre exècutable, vous pouvez rencontrer diffèrents problémes :

- Votre programme a un chemin faux en ce qui concerne l'endroit oú sont stockès les jeux de caractéres. (Par dèfaut '`/usr/local/mysql/share/mysql/charsets`'). Cela peut être rèparè en utilisant l'option `--character-sets-dir` du programme en question.
- Le jeu de caractéres est un jeu de caractéres multi-octets qui ne peut être chargè dynamiquement. Dans ce cas, vous devez recompiler le programme en incluant le support du jeu de caractére.
- Le jeu de caractéres est un jeu de caractéres dynamique, mais vous n'avez pas de fichier de configuration lui ètant associè. Dans ce cas, vous devez installer le fichier de configuration du jeu de caractéres á partir d'une nouvelle distribution MySQL.
- Votre fichier '`Index`' ne contient pas le nom du jeu de caractéres.

  ```
  ERROR 1105: File '/usr/local/share/mysql/charsets/?.conf' not found
  (Errcode: 2)
  ```

  Dans ce cas, vous devez soit obtenir un nouveau fichier `Index` ou ajouter á la main le nom du jeu de caractéres manquant.

Pour les tables `MyISAM`, vous pouvez vèrifier le nom du jeu de caractéres et son nombre associè d'une table avec `myisamchk -dvv nom_de_table`.

4.7 Scripts serveur MySQL et utilitaires

4.7.1 Prèsentation des scripts serveurs et des utilitaires

Tous les programmes MySQL prennent des options diffèrentes. Toutefois, tous les programmes MySQL disposent de l'option `--help` qui vous aidera a connaitre la liste compléte des diffèrentes options. Essayez par exemple `mysql --help`.

Vous pouvez modifier toutes les valeurs par dèfaut des programmes en les plaçant dans le fichier de configuration. Section 4.1.2 [Option files], page 198.

Voici la liste des programmes cotè serveur de MySQL :

`myisamchk`
: Un utilitaire pour dècrire, vèrifier, optimiser et rèparer les tables MySQL. Comme `myisamchk` a de trés nombreuses fonctions, ce programme dispose de son propre chapitre dans la documentation. Voir Chapitre 4 [MySQL Database Administration], page 192.

`make_binary_distribution`
: Prèpare une version exècutable d'un MySQL compilè. Il doivent être envoyès par FTP á '`/pub/mysql/Incoming`' sur `support.mysql.com` pour en faire profiter les autres utilisateurs MySQL.

`mysqlbug` Le script de rapport de bogue. Ce script doit toujours être utilisè lorsque vous devez envoyez un rapport de bogues aux listes de diffusion de MySQL.

`mysqld` Le dèmon SQL. Il doit toujours être en fonctionnement.

`mysql_install_db`
Crèe les tables de droits MySQL, avec les droits par dèfaut. Il est gènèralement exècutè une fois, lors de la premiére installation de MySQL.

4.7.2 `safe_mysqld`, le script pére de `mysqld`

`safe_mysqld` est la mèthode recommandèe pour dèmarrer un dèmon `mysqld` sous Unix. `safe_mysqld` ajoute des fonctionnalitès de sècuritè telles que le redèmarrage automatique lorsqu'une erreur survient et l'enregistrement d'informations d'exècution dans un fichier de log.

Si vous n'utilisez pas `--mysqld=#` ou `--mysqld-version=#` `safe_mysqld` va utiliser un programme appelè `mysqld-max` s'il existe. Sinon, `safe_mysqld` va dèmarrer le dèmon `mysqld`. Cela rend trés facile le test de `mysqld-max` au lieu de `mysqld`; copiez simplement `mysqld-max` á cotè de `mysqld` et il sera utilisè.

Normalement, vous ne devriez jamais èditer le script `safe_mysqld`, mais plutôt utiliser les options de `safe_mysqld` dans la section `[safe_mysqld]` du fichier '`my.cnf`'. `safe_mysqld` va lire toutes les options des sections `[mysqld]`, `[server]` et `[safe_mysqld]`, dans le fichier d'options. Voir Section 4.1.2 [Option files], page 198.

Notez que toutes les options de ligne de commande passèes á `safe_mysqld` sont transmises á `mysqld`. Si vous voulez utiliser une option de `safe_mysqld` que `mysqld` ne supporte pas, vous devez la spècifier dans le fichier d'options.

La plupart des options de `safe_mysqld` sont les mêmes que celles de `mysqld`. Voir Section 4.1.1 [Command-line options], page 192.

`safe_mysqld` supporte les options suivantes :

`--basedir=path`
`--core-file-size=#`
Taille du fichier core que `mysqld` doit être capable de crèer. Il est passè á `ulimit -c`.

`--datadir=path`
`--defaults-extra-file=path`
`--defaults-file=path`
`--err-log=path`
`--ledir=path`
Chemin de `mysqld`

`--log=path`
`--mysqld=mysqld-version`
Nom de la version de `mysqld` dans le dossier `ledir` que vous voulez dèmarrer.

`--mysqld-version=version`
Similaire á `--mysqld=` mais vous ne donnez que le suffixe de `mysqld`. Par exemple, si vous utilisez `--mysqld-version=max`, `safe_mysqld` va dèmarrer la version `ledir/mysqld-max`. Si l'argument de `--mysqld-version` est vide, `ledir/mysqld` sera utilisè.

`--no-defaults`
`--open-files-limit=#`
: Nombre de fichiers que `mysqld` doit être capable d'ouvrir. Passè á `ulimit -n`. Notez que vous devez dèmarrer `safe_mysqld` en tant que root pour que cette option fonctionne correctement !

`--pid-file=path`
`--port=#`

`--socket=path`
`--timezone=#`
: Configure le fuseau horaire (le `TZ`).

`--user=#`

Le script `safe_mysqld` a ètè ècrit pour qu'il soit capable de dèmarrer le serveur qui a ètè installè á partir des sources ou de la version binaire, même si l'installation de MySQL est lègérement exotique. `safe_mysqld` suppose que les conditions suivantes sont remplies :

- Le serveur et les bases de donnèes sont placèes dans un dossier relativement au dossier d'oú `safe_mysqld` est appelè. `safe_mysqld` cherche dans les sous dossiers '`bin`' et '`data`' (pour les distributions binaires) et, '`libexec`' et '`var`' (pour les distributions sources). Cette condition doit être remplie si vous exècutez `safe_mysqld` depuis votre dossier d'installation MySQL (par exemple, '`/usr/local/mysql`' pour une distribution binaire).
- Si le serveur et les bases de donnèes ne peuvent être trouvèes dans le dossier de travail, `safe_mysqld` essaie de les trouver en utilisant leurs chemins absolus. Les chemin typiquement ètudiès sont '`/usr/local/libexec`' et '`/usr/local/var`'. Les chemins rèels sont dèterminès lorsque la distribution est compilèe, et `safe_mysqld` a alors aussi ètè gènèrè. Ils doivent être corrects si MySQL a ètè installè dans un dossier standard.

Comme `safe_mysqld` essaie de trouver le serveur et les bases dans un dossier situè dans le dossier de travail, vous pouvez installer la version binaire de MySQL n'importe oú, du moment que vous dèmarrez le script `safe_mysqld` dans le dossier d'installation de MySQL :

```
shell> cd mysql_installation_directory
shell> bin/safe_mysqld &
```

Si `safe_mysqld` èchoue, même si il est appelè depuis le dossier d'installation, vous pouvez le modifier pour qu'il reconnaisse le chemin que vous utilisez jusqu'á `mysqld`. Notez que si vous faites èvoluer votre installation de MySQL, votre version de `safe_mysqld` sera ècrasèe, et vous devrez la rèèditer.

4.7.3 `mysqld_multi`, un programme pour gèrer plusieurs serveurs MySQL

`mysqld_multi` sert á gèrer plusieurs serveurs `mysqld` qui utilisent diffèrentes sockets Unix et ports TCP/IP.

Le programme va rechercher les groupes nommès `[mysqld#]` dans le fichier '`my.cnf`' (ou le fichier appelè `--config-file=...`), oú `#` peut être n'importe quel nombre positif, supèrieur

ou ègal á 1. Ce nombre est appelè le numèro de groupe d'options. Les numèros de groupe permettent de distinguer un groupe d'options d'un autre, et sont utilisès comme argument du script `mysqld_multi` pour spècifier quel serveur vous voulez dèmarrer, arrêter ou examiner. Les options listèes dans ces groupes doivent être les mêmes que celle que vous utiliseriez dans une section dèdièe au dèmon `[mysqld]`. Voyez, par exemple, Section 2.4.3 [Automatic start], page 104. Cependant, pour `mysqld_multi`, vous devez vous assurer que chaque groupe contient des valeurs pour les options telles que port, socket, etc., qui seront utilisèes par chaque processus `mysqld`.

`mysqld_multi` est utilisè ave la syntaxe suivante :

```
Usage: mysqld_multi [OPTIONS] {start|stop|report} [GNR,GNR,GNR...]
or     mysqld_multi [OPTIONS] {start|stop|report} [GNR-GNR,GNR,GNR-GNR,...]
```

Chaque GNR reprèsente un numèro de groupe d'options. Vous pouvez dèmarrer, arrêter ou examiner n'importe quel numèro de groupe d'options, ou même plusieurs d'entre eux en même temps. Pour un exemple de comment configurer le fichier d'options, utilisez cette commande :

```
shell> mysqld_multi --example
```

Les valeurs de numèro de groupe d'options peuvent être une liste de valeurs sèparèes par une virgule ou un tiret. Dans ce dernier cas, toutes les numèro de groupe d'options situès entre les deux numèros seront alors affectès. Sans numèro de groupe d'options spècifiè, tous les numèros de groupes du fichier d'options sont affectès. Notez que vous ne devez pas avoir d'espace dans la liste des numèros de groupe d'options. Tout ce qui est placè au-delá de l'espace sera ignorè.

`mysqld_multi` supporte les options suivantes :

`--config-file=...`
: Un fichier de configuration alternatif. Note : cela ne va pas modifier les options de ce programme (`[mysqld_multi]`), mais uniquement les groupes `[mysqld#]`. Sans cette option, tout sera lu dans le fichier d'options traditionnel '`my.cnf`'.

`--example`
: Affiche un exemple de fichier de configuration.

`--help`
: Affiche l'aide et quitte.

`--log=...`
: Fichier de log. Le chemin complet et le nom du fichier sont nècessaires. Note : si le fichier existe dèjá, les prochaines donnèes seront ajoutèes au fichier.

`--mysqladmin=...`
: L'exècutable `mysqladmin` á utiliser lors de l'arrêt du serveur.

`--mysqld=...`
: L'exècutable `mysqld` á utiliser. Notez que vous pouvez donner cette option á `safe_mysqld`. Ces options sont passèes á `mysqld`. Assurez-vous que vous avez bien `mysqld` dans votre variable d'environnement `PATH` ou corrigez `safe_mysqld`.

`--no-log`
: Affiche les donnèes d'historique á l'ècran plutôt que dans le fichier de log. Par dèfaut, le fichier de log est activè.

`--password=...`
: Le mot de passe de l'utilisateur `mysqladmin`.

`--tcp-ip` Connexion au serveur MySQL via le port TCP/IP au lieu de la socket Unix. Cela affecte l'arrêt et le rapport. Si le fichier de socket manque, le serveur peut continuer de tourner, mais il n'est plus accessible que par port TCP/IP. Par dèfaut, les connexions sont faites avec les sockets Unix.

`--user=...`
: L'utilisateur MySQL pour `mysqladmin`.

`--version`
: Affiche le numèro de version et quitte.

Quelques notes pour `mysqld_multi` :

- Assurez-vous que l'utilisateur MySQL, qui stoppe les services `mysqld` (e.g en utilisant la commande `mysqladmin`), a les mêmes nom d'utilisateur et mot de passe pour tous les dossiers de donnèes utilisès. Et assurez-vous que cet utilisateur a bien les droits de `SHUTDOWN`! Si vous avez de nombreux dossiers de donnèes et de nombreuses bases `mysql` avec diffèrents mots de passe pour le serveur `root` MySQL, vous souhaiterez peut être crèer un utilisateur commun `multi_admin` á chaque base, avec le même mot de passe (voir ci-dessous). Voici comment faire :

  ```
  shell> mysql -u root -S /tmp/mysql.sock -proot_password -e
  "GRANT SHUTDOWN ON *.* TO multi_admin@localhost IDENTIFIED BY 'multipass'"
  ```

  Voir Section 4.2.6 [Privileges], page 209. Vous devrez utiliser la même commande pour chaque serveur `mysqld` qui fonctionne : changez simplement la socket, `-S=...`).

- `pid-file` est trés important, si vous utilisez `safe_mysqld` pour dèmarrer `mysqld` (e.g., `--mysqld=safe_mysqld`). Chaque `mysqld` doit avoir son propre fichier `pid-file`. L'avantage d'utiliser `safe_mysqld` au lieu de `mysqld` est que `safe_mysqld` "surveille" tous les processus `mysqld` et les redèmarrera si un processus `mysqld` s'arrête suite á la reception d'un signal `kill -9`, ou pour toute autre raison comme une erreur de segmentation (que MySQL ne devrait jamais faire, bien sûr !). Notez bien que le script `safe_mysqld` vous imposera peut être d'être dèmarrè depuis un dossier spècial. Cela signifie que vous devrez probablement utiliser la commande shell `cd` jusqu'á un certain dossier avant de pouvoir exècuter `mysqld_multi`. Si vous avez des problémes pour dèmarrer, voyez le script `safe_mysqld`. Vèrifiez notamment ces lignes :

  ```
  ----------------------------------------------------------------------
  MY_PWD=`pwd` Check if we are starting this relative (for the binary
  release) if test -d /data/mysql -a -f ./share/mysql/english/errmsg.sys
  -a -x ./bin/mysqld
  ----------------------------------------------------------------------
  ```

  Voir Section 4.7.2 [`safe_mysqld`], page 292. Le test ci-dessus devrait fonctionner, ou bien vous rencontrerez probablement des problémes.

- N'oubliez jamais les problémes que peut soulever le dèmarrage de plusieurs dèmons `mysqld` avec le même dossier de donnèes. Utilisez des dossiers de donnèes sèparès, á moins que vous ne **sachiez bien** ce que vous faites.

- Le fichier de socket et le port TCP/IP doivent être diffèrents pour chaque `mysqld`.

- Le premier et le cinquiéme groupe `mysqld` on ètè intentionnellement ignorès dans le groupe d'exemple. Vous pouvez laisser des trous dans le fichier de configuration. Cela vous donnera plus de souplesse. L'ordre dans lequel les dèmons `mysqlds` sont dèmarrès ou arrètès dèpend de l'ordre dans lequel ils apparaissent dans le fichier de configuration.
- Lorsque vous voulez faire rèfèrence á un groupe en utilisant le numèro de groupe d'options, utilisez simplement le numèro du groupe á la fin du nom du groupe. Par exemple, le numèro de groupe de `[mysqld17]` est le 17.
- Vous pouvez vouloir utiliser l'option `--user` pour `mysqld`, mais afin de faire cela, vous devez exècuter le script `mysqld_multi` en tant que `root` Unix. Placer cette option dans le fichier de configuration ne changera rien : vous obtiendrez une alerte, si vous n'êtes pas le super utilisateur, et les dèmons `mysqld` seront dèmarrès avec vos droits Unix. **Important** : assurez-vous bien que le fichier de donnèes et le fichier de `pid-file` sont accessibles en lecture et ècriture (et exècution pour le dernier) á l'utilisateur Unix qui lance les processus `mysqld`. **N'utilisez pas** le compte root Unix pour cela, á moins que vous ne **sachiez** ce que vous faîtes.
- **Trés important** : assurez-vous de bien comprendre la signification des options que vous passez á `mysqld`s et **pourquoi vous avez besoin** de plusieurs processus `mysqld`. Dèmarrer plusieurs serveurs `mysqld`s dans le même dossier **ne vous donnera aucun** gain de performance dans un systéme threadè.

Voir Section 4.1.4 [Multiple servers], page 202.

Voici un exemple de fichier de configuration fourni par `mysqld_multi`.

```
# This file should probably be in your home dir (~/.my.cnf) or /etc/my.cnf
# Version 2.1 by Jani Tolonen

[mysqld_multi]
mysqld     = /usr/local/bin/safe_mysqld
mysqladmin = /usr/local/bin/mysqladmin
user       = multi_admin
password   = multipass

[mysqld2]
socket     = /tmp/mysql.sock2
port       = 3307
pid-file   = /usr/local/mysql/var2/hostname.pid2
datadir    = /usr/local/mysql/var2
language   = /usr/local/share/mysql/english
user       = john

[mysqld3]
socket     = /tmp/mysql.sock3
port       = 3308
pid-file   = /usr/local/mysql/var3/hostname.pid3
datadir    = /usr/local/mysql/var3
language   = /usr/local/share/mysql/swedish
user       = monty
```

```
[mysqld4]
socket     = /tmp/mysql.sock4
port       = 3309
pid-file   = /usr/local/mysql/var4/hostname.pid4
datadir    = /usr/local/mysql/var4
language   = /usr/local/share/mysql/estonia
user       = tonu

[mysqld6]
socket     = /tmp/mysql.sock6
port       = 3311
pid-file   = /usr/local/mysql/var6/hostname.pid6
datadir    = /usr/local/mysql/var6
language   = /usr/local/share/mysql/japanese
user       = jani
```

Voir Section 4.1.2 [Option files], page 198.

4.7.4 myisampack, le gènèrateur de tables MySQL compressèes en lecture seule

`myisampack` sert á compresser des tables MyISAM et `pack_isam` sert á compresser les tables ISAM. Comme les tables ISAM sont obsolétes, nous ne traiterons que de `myisampack`, mais tout ce qui est dit au sujet de `myisampack` est aussi vrai pour `pack_isam`.

`myisampack` fonctionne en compressant sèparèment chaque colonne de la table. Les informations nècessaires á la dècompression sont lues en mèmoire lorsque la table est ouverte. Cela donne de bien meilleures performances lors de l'accés á des lignes individuelles, car nous n'avez qu'á dècompresser exactement une des lignes, et non pas un bloc de disque entier. Gènèralement, `myisampack` compresse le fichier avec un gain de 40 á 70 %.

MySQL utilise la carte mèmoire (`mmap()`) sur les tables compressèes et utilise les outils classiques de lecture et ècriture si `mmap()` ne fonctionne pas.

Notez bien ceci :

- Aprés avoir compressè la table, celle-ci n'est plus accessible qu'en lecture. C'est souvent un ètat voulu (par exemple, pour être gravèe sur un CD). De plus, autoriser les ècritures dans une table compressèe fait partie de notre liste de tâche, mais avec une trés faible prioritè.
- `myisampack` peut aussi compresser des colonnes `BLOB` ou `TEXT`. L'ancien `pack_isam` (pour les tables `ISAM`) ne peut le faire.

`myisampack` est invoquè comme ceci :

```
shell> myisampack [options] filename ...
```

Chaque nom de fichier doit être le nom d'un fichier d'index ('`.MYI`'). Si vous n'êtes pas dans le dossier de donnèes, vous devez spècifier le chemin complet jusqu'au fichier. Il est tolèrè que vous omettiez l'extension du fichier '`.MYI`'.

`myisampack` supporte les options suivantes :

`-b, --backup`
: Fait une sauvegarde de la table sous le nom de `tbl_name.OLD`.

`-#, --debug=debug_options`
: Affiche le log de dèbogage. La chaîne `debug_options` vaut souvent `'d:t:o,filename'`.

`-f, --force`
: Force la compression de la table, même si elle grossit ou si le fichier temporaire existe dèjá. `myisampack` crèe un fichier temporaire appelè '`tbl_name.TMD`' lors de la compression. Si vous tuez `myisampack`, le fichier '`.TMD`' peut ne pas être effacè. Normalement, `myisampack` se termine avec une erreur s'il dècouvre que le fichier '`tbl_name.TMD`' existe. Avec `--force`, `myisampack` reprendra le travail.

`-?, --help`
: Affiche le message d'aide et quitte.

`-j big_tbl_name, --join=big_tbl_name`
: Rassemble toutes les tables indiquèes dans la ligne de commande dans une seule table appelèe `big_tbl_name`. Toutes les tables qui seront combinèes **doivent** être identiques (mêmes noms de colonnes, mêmes types, mêmes index, etc.)

`-p #, --packlength=#`
: Spècifie la taille de stockage de la longueur de ligne, en octets. Cette valeur doit être 1, 2, ou 3. (`myisampack` stocke toutes les lignes avec des pointeurs de lignes de 1, 2 ou 3 octets. Dans les cas normaux, `myisampack` peut dèterminer la taille correcte avant de compresser le fichier, mais il peut aussi se rendre compte durant le processus qu'une autre taille aurait ètè plus approprièe, ou plus courte. Dans ce cas, `myisampack` va imprimer une note pour que vous le sachiez lors de la prochaine compression du même fichier.

`-s, --silent`
: Mode silencieux. Seules les erreurs seront affichèes.

`-t, --test`
: Ne compresse pas la table, mais teste juste la compression.

`-T dir_name, --tmp_dir=dir_name`
: Utilise le dossier indiquè comme dossier pour les fichiers temporaires.

`-v, --verbose`
: Mode dètaillè. Toutes les informations sur la progression de la compression seront affichèes.

`-V, --version`
: Affiche la version et quitte.

`-w, --wait`
: Attend et reessaie, si la table ètait dèjá en cours d'utilisation. Si le serveur `mysqld` a ètè dèmarrè avec l'option `--skip-external-locking`, ce n'est pas une bonne idèe d'appeler `myisampack`, car la table risque d'être modifièe durant la compression.

La sèquence de commande illustre la session de compression :

```
shell> ls -l station.*
-rw-rw-r--   1 monty    my         994128 Apr 17 19:00 station.MYD
-rw-rw-r--   1 monty    my          53248 Apr 17 19:00 station.MYI
-rw-rw-r--   1 monty    my           5767 Apr 17 19:00 station.frm

shell> myisamchk -dvv station

MyISAM file:     station
Isam-version:  2
Creation time: 1996-03-13 10:08:58
Recover time:  1997-02-02  3:06:43
Data records:              1192  Deleted blocks:              0
Datafile: Parts:           1192  Deleted data:                0
Datafile pointer (bytes):     2  Keyfile pointer (bytes):     2
Max datafile length:   54657023  Max keyfile length:   33554431
Recordlength:               834
Record format: Fixed length

table description:
Key Start Len Index   Type                     Root  Blocksize    Rec/key
1   2     4   unique  unsigned long             1024       1024          1
2   32    30  multip. text                     10240       1024          1

Field Start Length Type
1     1     1
2     2     4
3     6     4
4     10    1
5     11    20
6     31    1
7     32    30
8     62    35
9     97    35
10    132   35
11    167   4
12    171   16
13    187   35
14    222   4
15    226   16
16    242   20
17    262   20
18    282   20
19    302   30
20    332   4
21    336   4
```

```
22    340   1
23    341   8
24    349   8
25    357   8
26    365   2
27    367   2
28    369   4
29    373   4
30    377   1
31    378   2
32    380   8
33    388   4
34    392   4
35    396   4
36    400   4
37    404   1
38    405   4
39    409   4
40    413   4
41    417   4
42    421   4
43    425   4
44    429   20
45    449   30
46    479   1
47    480   1
48    481   79
49    560   79
50    639   79
51    718   79
52    797   8
53    805   1
54    806   1
55    807   20
56    827   4
57    831   4

shell> myisampack station.MYI
Compressing station.MYI: (1192 records)
- Calculating statistics

normal:     20  empty-space:      16  empty-zero:        12  empty-fill:  11
pre-space:   0  end-space:        12  table-lookups:      5  zero:         7
Original trees:  57  After join: 17
- Compressing file
87.14%
```

```
shell> ls -l station.*
-rw-rw-r--   1 monty    my         127874 Apr 17 19:00 station.MYD
-rw-rw-r--   1 monty    my          55296 Apr 17 19:04 station.MYI
-rw-rw-r--   1 monty    my           5767 Apr 17 19:00 station.frm

shell> myisamchk -dvv station

MyISAM file:     station
Isam-version:  2
Creation time: 1996-03-13 10:08:58
Recover time:  1997-04-17 19:04:26
Data records:              1192  Deleted blocks:              0
Datafile: Parts:           1192  Deleted data:                0
Datafilepointer (bytes):      3  Keyfile pointer (bytes):     1
Max datafile length:   16777215  Max keyfile length:     131071
Recordlength:               834
Record format: Compressed

table description:
Key Start Len Index   Type                         Root  Blocksize    Rec/key
1   2     4   unique  unsigned long               10240       1024          1
2   32    30  multip. text                        54272       1024          1

Field Start Length Type                          Huff tree  Bits
1     1     1      constant                              1     0
2     2     4      zerofill(1)                           2     9
3     6     4      no zeros, zerofill(1)                 2     9
4     10    1                                            3     9
5     11    20     table-lookup                          4     0
6     31    1                                            3     9
7     32    30     no endspace, not_always               5     9
8     62    35     no endspace, not_always, no empty     6     9
9     97    35     no empty                              7     9
10    132   35     no endspace, not_always, no empty     6     9
11    167   4      zerofill(1)                           2     9
12    171   16     no endspace, not_always, no empty     5     9
13    187   35     no endspace, not_always, no empty     6     9
14    222   4      zerofill(1)                           2     9
15    226   16     no endspace, not_always, no empty     5     9
16    242   20     no endspace, not_always               8     9
17    262   20     no endspace, no empty                 8     9
18    282   20     no endspace, no empty                 5     9
19    302   30     no endspace, no empty                 6     9
20    332   4      always zero                           2     9
21    336   4      always zero                           2     9
22    340   1                                            3     9
23    341   8      table-lookup                          9     0
```

```
24    349   8      table-lookup                    10     0
25    357   8      always zero                      2     9
26    365   2                                       2     9
27    367   2      no zeros, zerofill(1)            2     9
28    369   4      no zeros, zerofill(1)            2     9
29    373   4      table-lookup                    11     0
30    377   1                                       3     9
31    378   2      no zeros, zerofill(1)            2     9
32    380   8      no zeros                         2     9
33    388   4      always zero                      2     9
34    392   4      table-lookup                    12     0
35    396   4      no zeros, zerofill(1)           13     9
36    400   4      no zeros, zerofill(1)            2     9
37    404   1                                       2     9
38    405   4      no zeros                         2     9
39    409   4      always zero                      2     9
40    413   4      no zeros                         2     9
41    417   4      always zero                      2     9
42    421   4      no zeros                         2     9
43    425   4      always zero                      2     9
44    429   20     no empty                         3     9
45    449   30     no empty                         3     9
46    479   1                                      14     4
47    480   1                                      14     4
48    481   79     no endspace, no empty           15     9
49    560   79     no empty                         2     9
50    639   79     no empty                         2     9
51    718   79     no endspace                     16     9
52    797   8      no empty                         2     9
53    805   1                                      17     1
54    806   1                                       3     9
55    807   20     no empty                         3     9
56    827   4      no zeros, zerofill(2)            2     9
57    831   4      no zeros, zerofill(1)            2     9
```

Les informations affichèes par `myisampack` sont dècrites ici :

`normal` Le nombre de colonnes pour lesquelles aucune compression n'est utilisèe.

`empty-space`
: Le nombre de colonnes dont les valeurs ne contiennent que des octets : elles n'occuperont plus qu'un octet.

`empty-zero`
: Le nombre de colonnes dont les valeurs ne contiennent que des zèros : elles n'occuperont plus qu'un octet.

`empty-fill`
: Le nombre de colonnes de type entier qui n'occupent pas la totalitè de l'espace de leur type. Elles seront rèduites en taille (par exemple, une colonne de type `INTEGER` sera transformèe en `MEDIUMINT`).

`pre-space`
: Le nombre de colonnes de nombres á virgule flottante qui ont des valeurs stockèes avec des espaces initiaux. Dans ce cas, chaque valeur va contenir le nombre d'espace initiaux.

`end-space`
: Le nombre de colonnes qui ont de nombreux espaces terminaux. Dans ce cas, chaque valeur va contenir un compte du nombre d'espaces terminaux.

`table-lookup`
: La colonne n'a que quelques valeurs diffèrentes, qui seront converties en une colonne de type `ENUM` avant une compression de type Huffman.

`zero` Le nombre de colonnes pour lesquelles toutes les valeurs sont zèro.

`Original trees`
: Le nombre initial d'arbres Huffman.

`After join`
: Le nombre d'arbres Huffman distincts obtenus aprés avoir joint les arbres pour èconomiser de l'espace d'entête.

Aprés la compression d'une table, `myisamchk -dvv` affiche des informations supplèmentaires pour chaque champ :

`Type` Le type de fichier peut contenir les informations suivantes :

`constant` Toutes les lignes ont la même valeur.

`no endspace`
: Ne stocke pas les espaces finaux.

`no endspace, not_always`
: Ne stocke pas les espaces finaux et ne compresse pas les espaces finaux pour toutes les valeurs.

`no endspace, no empty`
: Ne stocke pas les espaces finaux. Ne stocke pas les valeurs vides.

`table-lookup`
: La colonne a ètè convertie en `ENUM`.

`zerofill(n)`
: Les `n` chiffres significatifs sont toujours 0, et n'ont pas ètè stockès.

`no zeros` Ne stocke pas les zèros.

`always zero`
: Les valeurs 0 sont stockèes sur un octet.

`Huff tree` L'arbre Huffman associè au champ.

`Bits` Le nombre de bits utilisès par l'arbre Huffman.

Aprés la compression de `pack_isam`/`myisampack` vous devez exècuter la commande `isamchk`/`myisamchk` pour recrèer l'index. A ce moment, vous pouvez aussi trier les blocs d'index et crèer des statistiques nècessaires pour l'optimisateur MySQL :

```
myisamchk -rq --analyze --sort-index table_name.MYI
isamchk   -rq --analyze --sort-index table_name.ISM
```

Aprés avoir installè la table compressèe dans un dossier de donnèes MysQL, vous devez exècutez la commande `mysqladmin flush-tables` pour forcer `mysqld` á utiliser cette nouvelle table.

Si vous voulez dècompresser une table compressèe, vous pouvez le faire avec l'option `--unpack` de la commande `isamchk` ou `myisamchk`.

4.7.5 `mysqld-max`, la version ètendue du serveur `mysqld`

`mysqld-max` est le serveur MySQL (`mysqld`) configurè avec les options suivantes :

Option	Comment
–with-server-suffix=-max	Ajoute un suffixe á la chaîne de version de `mysqld`
–with-innodb	Supporte les tables InnoDB.
–with-bdb	Supporte les tables Berkeley DB (BDB)
CFLAGS=-DUSE_SYMDIR	Support des liens symboliques sous Windows.

Vous pouvez obtenir les exècutables MySQL-max á `http://www.mysql.com/downloads/mysql-max-3.23.`

La distribution Windows de MySQL inclut les deux programmes standard `mysqld.exe` et `mysqld-max.exe`. `http://www.mysql.com/downloads/mysql-3.23.html`. Voir Section 2.1.2 [Windows installation], page 70.

Notez que les tables InnoDB et Berkeley DB ne sont pas disponibles sur toutes les plateformes, et que certaines versions `Max` ne supportent pas les deux. Vous pouvez vèrifier quelles tables sont supportèes avec la requête suivante :

```
mysql> SHOW VARIABLES LIKE "have_%";
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| have_bdb      | YES   |
| have_innodb   | NO    |
| have_isam     | YES   |
| have_raid     | NO    |
| have_openssl  | NO    |
+---------------+-------+
```

La signification des valeurs est :

Valeur	Signification
`YES`	L'option est activatèe et utilisable.
`NO`	MySQL n'a pas ètè compilè avec le support pour cette option.

`DISABLED` L'option xxxx est dèsactivèe car `mysqld` a ètè lancè avec `--skip-xxxx` ou parce que `mysqld` n'a pas ètè lancè avec les bonnes options. Dans ce cas, le fichier `hostname.err` devrait contenir la raison pour laquelle l'option est dèsactivèe.

Note : pour être capable de crèer des tables InnoDB, vous **devez** èditer vos options de dèmarrage et inclure au moins l'option `innodb_data_file_path`. Voir Section 7.5.2 [InnoDB start], page 545.

Pour avoir de meilleures performances avec les tables BDB, vous devriez aussi ajouter certaines options de configuration. Voir Section 7.6.3 [BDB start], page 586.

`safe_mysqld` va automatiquement lancer tout programme `mysqld` avec le suffixe `-max`. Cela rend trés facile les tests de `mysqld`, dans le cadre d'un installation existante. Il suffit d'exècuter le programme `configure` avec les options que vous souhaitez, et d'installer un nouveau programme `mysqld` sous le nom de `mysqld-max` dans le même dossier que votre ancien `mysqld`. Voir Section 4.7.2 [`safe_mysqld`], page 292.

Les paquets RPM `mysqld-max` utilisent la fonctionnalitè de `safe_mysqld` mentionnèe plus tôt. Il suffit d'installer le programme `mysqld-max` et `safe_mysqld` va automatiquement utiliser cet exècutable lorsque `safe_mysqld` sera redèmarrè.

La table suivante illustre les types de tables que MySQL-Max inclut :

Systéme	`BDB`	`InnoDB`
AIX 4.3	Non	Oui
HP-UX 11.0	Non	Oui
Linux-Alpha	Non	Oui
Linux-Intel	Oui	Oui
Linux-IA64	Non	Oui
Solaris-Intel	Non	Oui
Solaris-SPARC	Oui	Oui
Caldera	Oui	Oui
(SCO) OSR5		
UnixWare	Oui	Oui
Windows/NT	Oui	Oui

4.8 MySQL Scripts clients et utilitaires

4.8.1 Prèsentation des scripts serveurs et utilitaires

Tous les clients MySQL qui communiquent avec le serveur via la librairie `mysqlclient` utilisent les variables d'environnement suivantes :

Nom	**Description**
`MYSQL_UNIX_PORT`	La socket par dèfaut; utilisè pour les connexions á `localhost`
`MYSQL_TCP_PORT`	Le port TCP/IP par dèfaut
`MYSQL_PWD`	Le mot de passe par dèfaut
`MYSQL_DEBUG`	Des options de dèbogage et de traçage

`TMPDIR` Le dossier oú les fichiers et tables temporaires sont èrits

L'utilisation de `MYSQL_PWD` n'est pas une technique sûre. Voir Section 4.2.8 [Connecting], page 215.

Le client '`mysql`' utilise le fichier indiquè dans la variable d'environnement `MYSQL_HISTFILE` pour sauver l'historique de la ligne de commande. La valeur par dèfaut pour la situation du fichier d'historique est '`$HOME/.mysql_history`', oú `$HOME` est la valeur de la variable d'environnement `HOME`. Voir Annexe F [Environment variables], page 828.

Tous les programmes MySQL utilisent diffèrentes options. Toutefois, tous les programmes MySQL disposent de l'option `--help` pour vous donner une description compléte de toutes les options du programme. Par exemple, essayez d'utiliser `mysql --help`.

Vous pouvez remplacer les valeurs par dèfaut des options de tous les clients standard en mettant d'autres valeurs dans le fichier de configuration. Section 4.1.2 [Option files], page 198.

Voici un bref aperçu des programmes clients MySQL :

`msql2mysql`
: Un script shell qui convertit les programmes `mSQL` vers MySQL. Il ne gére pas toutes les situations, mais c'est une trés bonne base de travail.

`mysqlaccess`
: Un script qui vèrifie les droits d'accés du trio hôte, utilisateur et base de donnèes.

`mysqladmin`
: Un utilitaire pour rèaliser des opèrations d'administration de la base, telles que les crèations de bases, le rafraîchissement des tables de droits, l'ècriture des tables sur le disque et la rèouverture des fichiers de log. á `mysqladmin` permet aussi de lire la version, les processus et les informations de statut du serveur. Voir Section 4.8.3 [`mysqladmin`], page 314.

`mysqldump`
: Exporte une base de donnèes MySQL dans un fichier sous la forme de requêtes SQL, ou de fichiers texte, avec la tabulation comme sèparateur. Un freeware amèliorè, d'aprés une idèe originale de Igor Romanenko. Voir Section 4.8.5 [`mysqldump`], page 318.

`mysqlimport`
: Importe les fichiers textes dans les tables, en utilisant la commande `LOAD DATA INFILE`. Voir Section 4.8.7 [`mysqlimport`], page 323.

`mysqlshow`
: Affiche des informations sur les bases, tables, colonnes et index.

`replace` Un utilitaire qui est utilisè par `msql2mysql`, qui a d'autres applications gènèrales. `replace` modifie des chaînes dans des fichiers, ou sur l'entrèe standard. Utilise une machine d'ètats pour rechercher les plus grands chaînes en premier. Sert á èchanger des chaînes. Par exemple, cette commande èchange les chaînes `a` et `b` dans les fichiers spècifiès :

```
shell> replace a b b a -- file1 file2 ...
```

4.8.2 `mysql`, The Command-line Tool

`mysql` est un simple script SQL (qui exploite GNU `readline`). Il supporte une utilisation interactive et non-interactive. Lorsqu'il est utilisè interactivement, les rèsultats des requêtes sont prèsentès sous la forme d'une table au format ASCII. Lorsqu'il est utilisè non-interactivement, par exemple, comme filtre, le rèsultat est fourni au format de liste avec sèparation par tabulation (le format d'affichage peut être modifiè en utilisant les options de ligne de commande). Vous pouvez exècuter le script comme ceci :

```
shell> mysql database < script.sql > output.tab
```

Si vous avez des problémes liès á des insuffisances de mèmoire avec le client, utilisez l'option `--quick`! Cela force `mysql` á utiliser `mysql_use_result()` plutôt que `mysql_store_result()` pour lire les rèsultats.

Utiliser `mysql` est trés simple. Il suffit de le dèmarrer comme ceci : `mysql database` ou `mysql --user=user_name --password=your_password database`. Tapez une commande SQL, puis terminez-la avec '`;`', '`\g`' ou '`\G`', et finissez avec Entrèe.

`mysql` supports the following options:

`-?, --help`
: Affiche cette aide et quitte.

`-A, --no-auto-rehash`
: Pas de rehachage automatique. Il faut utiliser la commande 'rehash' pour obtenir la complètion des noms de tables et champs. Cela accèlére le dèmarrage du client.

`--prompt=...`
: Modifie l'invite de commande de MySQL.

`-b, --no-beep`
: Eteind le son d'erreur.

`-B, --batch`
: Affiche les rèsultats avec une tabulation comme rèsultat, et chaque ligne avec une nouvelle ligne. N'utilise pas l'historique.

`--character-sets-dir=...`
: Le dossier oú les jeux de caractéres sont crèès.

`-C, --compress`
: Utilise la compression avec le protocole client-serveur.

`-#, --debug[=...]`
: Dèmarre le log de dèbogage. Par dèfaut, il vaut 'd:t:o,/tmp/mysql.trace'.

`-D, --database=...`
: La base de donnèes á utiliser. C'est particuliérement pratique dans le fichier d'options '`my.cnf`'.

`--default-character-set=...`
: Configure le jeu de caractéres par dèfaut.

`-e, --execute=...`
Exècute une commande et quitte. Le rèsultat est au format de l'option –batch

`-E, --vertical`
Affiche le rèsultat d'une requête verticalement. Sans cette option, vous pouvez aussi obtenir ce format en terminant votre requête avec `\G`.

`-f, --force`
Continue même si vous obtenez une erreur SQL.

`-g, --no-named-commands`
Les commandes nommèes sont dèsactivèes. Utilisez la forme * uniquement, ou utilisez les commandes nommèes au dèbut d'une ligne se terminant par un point-virgule (';'). Depuis la version 10.9, le client dèmarre avec cette option **activèe** par dèfaut. Avec l'option -g, le format long des commandes va continuer á fonctionner.

`-G, --enable-named-commands`
Les commandes nommèes sont **activèes**. Le format long est autorisè, ainsi que les commandes courtes *.

`-i, --ignore-space`
Ignore les espaces aprés les noms de fonctions.

`-h, --host=...`
Connexion avec l'hôte indiquè.

`-H, --html`
Produit un rèsultat au format HTML.

`-X, --xml` Produit un rèsultat au format XML.

`-L, --skip-line-numbers`
N'ècrit pas les numèros de lignes dans les erreurs. Trés pratique lorsque vous voulez comparer des rèsultats qui incluent des messages d'erreurs.

`--no-pager`
Dèsactive le systéme de page, et affiche directement dans la sortie standard. Voyez l'aide interactive (\h).

`--no-tee` Dèsactive le outfile. Voyez l'aide interactive (\h).

`-n, --unbuffered`
Vide le buffer de requête aprés chaque requête.

`-N, --skip-column-names`
N'ècrit pas les noms de colonnes dans les rèsultats.

`-O, --set-variable var=option`
Spècifie la valeur d'une variable. `--help` liste toutes ces variables. Notez bien que `--set-variable` est obsoléte depuis MySQL 4.0, et qu'il suffit dèsormais d'utiliser `--var=option`.

`-o, --one-database`
Ne modifie que la base par dèfaut. C'est pratique pour èviter les modifications dans les autres bases dans le fichier de log.

`--pager[=...]`
: Type d'affichage. Par dèfaut, la variable d'environnement `ENV` vaut `PAGER`. Les paginateurs valides sont less, more, cat [> filename], etc. Voyez l'aide interactive (\h). Cette option n'est pas effective en mode batch. Les paginateurs ne fonctionnent qu'avec Unix.

`-p[password], --password[=...]`
: Le mot de passe utilisè lors de la connexion sur le serveur. S'il n'est pas donnè en ligne de commande, il sera demandè interactivement. Notez que si vous utilisez la forme courte `-p`, vous ne devez pas laisser d'espace entre l'option et le mot de passe.

`-P --port=...`
: Le numèro de port TCP/IP pour la connexion.

`-q, --quick`
: Ne met pas en cache le rèsultat, et l'affiche ligne par ligne. C'est plus lent pour le serveur, si le rèsultat est interrompu. N'utilise pas le fichier d'historique.

`-r, --raw` Ecrit les valeurs des colonnes sans les conversions de protections. Utilisè en mode `--batch`

`-s, --silent`
: Mode trés silencieux.

`-S --socket=...`
: Le fichier de socket á utiliser pour la connexion.

`-t --table`
: Affichage au format de table. C'est le mode par dèfaut pour le mode non-batch.

`-T, --debug-info`
: Affiche des informations de dèbogage au moment de la fin du programme.

`--tee=...`
: Ajoute tout dans le fichier de sortie. Voyez l'aide interactive (\h). Ne fonctionne pas en mode batch.

`-u, --user=#`
: Nom d'utilisateur pour la connexion, si ce n'est pas l'utilisateur Unix courant.

`-U, --safe-updates[=#], --i-am-a-dummy[=#]`
: N'autorise que les commandes `UPDATE` et `DELETE` qui utilisent des clès. Voir plus bas pour des informations sur cette option. Vous pouvez annuler cette option si vous l'avez dans le fichier d'option '`my.cnf`' en utilisant la syntaxe `--safe-updates=0`.

`-v, --verbose`
: Affichage plus dètaillè (-v -v -v indique le format d'affichage de table).

`-V, --version`
: Affiche la version et quitte.

`-w, --wait`
: Attend et retente si la connexion s'interrompt, au lieu de quitter.

Vous pouvez aussi spècifier les variables suivantes avec l'option `-O` ou `--set-variable`; notez bien que `--set-variable` est obsoléte depuis MySQL 4.0, utilisez la nouvelle syntaxe `--var=option` :

Nom de variable	Par dèfaut	Description
connect_timeout	0	Nombre de secondes avant que la connexion n'expire.
max_allowed_packet	16777216	Taille maximale du paquet de communication avec le serveur.
net_buffer_length	16384	Buffer pour les communications TCP/IP et socket.
select_limit	1000	Limite automatique pour les commandes SELECT avec l'option –i-am-a-dummy
max_join_size	1000000	Limite automatique pour les commandes de jointure avec l'option –i-am-a-dummy.

Si vous tapez 'help' en ligne de commande, `mysql` va afficher les commandes qu'il supporte :

```
mysql> help

MySQL commands:
help    (\h)    Display this text.
?       (\h)    Synonym for 'help'.
clear   (\c)    Clear command.
connect (\r)    Reconnect to the server.
                Optional arguments are db and host.
edit    (\e)    Edit command with $EDITOR.
ego     (\G)    Send command to mysql server,
                display result vertically.
exit    (\q)    Exit mysql. Same as quit.
go      (\g)    Send command to mysql server.
nopager (\n)    Disable pager, print to stdout.
notee   (\t)    Don't write into outfile.
pager   (\P)    Set PAGER [to_pager].
                Print the query results via PAGER.
print   (\p)    Print current command.
prompt  (\R)    Change your mysql prompt.
quit    (\q)    Quit mysql.
rehash  (\#)    Rebuild completion hash.
source  (\.)    Execute a SQL script file.
                Takes a file name as an argument.
status  (\s)    Get status information from the server.
tee     (\T)    Set outfile [to_outfile].
                Append everything into given outfile.
use     (\u)    Use another database.
                Takes database name as argument.
```

La commande `pager` ne fonctionne que sous Unix.

La commande `status` donne des dètails sur la connexion et le serveur utilisès. Si vous fonctionnez en mode `--safe-updates`, `status` va aussi afficher les valeurs des variables de `mysql` qui affectent vos requêtes.

Une option de dèmarrage utile pour les dèbutants (introduit en MySQL version 3.23.11) est `--safe-updates` (ou `--i-am-a-dummy` pour les utilisateurs qui ont dèjá utilisè une commande `DELETE FROM table_name` mais on oubliè la clause `WHERE`. Lorsque vous utilisez cette option, `mysql` envoie la commande suivante au serveur MySQL lors de l'ouverture de la connexion :

```
SET SQL_SAFE_UPDATES=1,SQL_SELECT_LIMIT=#select_limit#,
    SQL_MAX_JOIN_SIZE=#max_join_size#"
```

oú `#select_limit#` et `#max_join_size#` sont des variables qui peuvent être configurèes depuis `mysql`. Voir Section 5.5.6 [SET OPTION], page 396.

L'effet de la commande ci-dessus est :

- Vous n'êtes pas autorisè á faire une commande `UPDATE` ou `DELETE` si vous n'avez pas de contrainte de clè dans la clause `WHERE`. Il est possible toutefois de forcer la commande `UPDATE/DELETE` en utilisant `LIMIT` :

  ```
  UPDATE table_name SET not_key_column=# WHERE not_key_column=# LIMIT 1;
  ```

- Tous les grands rèsultats sont automatiquement limitès á `#select_limit#` lignes.
- Les commandes `SELECT` qui vont probablement devoir examiner plus de `#max_join_size` lignes seront annulèes.

Quelques conseils pratiques sur le client `mysql` :

Certaines donnèes seront plus faciles á lire lorsqu'affichèes verticalement, au lieu d'horizontalement. Par exemple, des textes longs qui incluent des nouvelles lignes. Il sera alors plus facile de les lire verticalement.

```
mysql> SELECT * FROM mails WHERE LENGTH(txt) < 300 lIMIT 300,1\G
*************************** 1. row ***************************
  msg_nro: 3068
     date: 2000-03-01 23:29:50
time_zone: +0200
mail_from: Monty
    reply: monty@no.spam.com
  mail_to: "Thimble Smith" <tim@no.spam.com>
      sbj: UTF-8
      txt: >>>>> "Thimble" == Thimble Smith writes:

Thimble> Hi.  I think this is a good idea.  Is anyone familiar with UTF-8
Thimble> or Unicode? Otherwise, I'll put this on my TODO list and see what
Thimble> happens.

Yes, please do that.

Regards,
Monty
     file: inbox-jani-1
     hash: 190402944
1 row in set (0.09 sec)
```

Pour les logs, vous pouvez utiliser l'option `tee`. Le `tee` peut être dèmarrè avec l'option `--tee=...`, ou depuis la ligne de commande interactive avec la commande `tee`. Toutes les

données affichèes sur l'ècran seront enregistrèes dans un fichier spècifiè. Cela peut être trés utile pour pouvoir dèboguer. Le `tee` peut être dèsactivè avec la commande en ligne `notee`. Exècuter `tee` plusieurs fois va redèmarrer l'enregistrement. Sans paramétre, la commande va ouvrir le dernier fichier utilisè. Notez que `tee` va ècrire les donnèes aprés chaque requête, avant que l'invite apparaisse, pour attendre la prochaine commande.

Afficher les rèsultats en mode interactif avec les programmes Unix tels que less, more ou tout autre programme similaire est actuellement possible avec l'option `--pager[=...]`. Sans argument, le client `mysql` va rechercher la variable d'environnement PAGER et l'utiliser pour `pager`. `pager` peut être dèmarrè en ligne de commande avec `pager` et dèsactivèe avec `nopager`. La commande prend un argument optionnel et `pager` prendra alors cette valeur. La commande `pager` peut être appelèe sans argument, mais cela impose que l'option `--pager` ait ètè utilisèe, ou que le mode `pager` sera par dèfaut la sortie standard. `pager` fonctionne uniquement sous Unix, car il utilise la fonction popen(), qui n'existe pas sous Windows. Sous Windows, l'option `tee` peut être utilisèe á la place, même si elle n'est pas aussi pratique que `pager` dans certaines situations.

Quelques conseils sur `pager` :

- Vous pouvez l'utiliser pour ècrire dans un fichier :

  ```
  mysql> pager cat > /tmp/log.txt
  ```

  et les rèsultats iront directement dans le fichier. Vous pouvez aussi passer des options pour le programme que vous voulez utiliser comme `pager` :

  ```
  mysql> pager less -n -i -S
  ```

- Dans la prèsentation ci-dessus, notez bien l'option '-S'. Vous pourriez la trouver trés pratique lorsque vous ètudiez des rèsultats : essayer l'option d'affichage horizontal (validez vos commandes avec '\g' ou ';') et vertical (terminez vos commandes avec '\G'). Parfois, un rèsultat trés large est difficile á lire et avec l'option -S dirigèe vers less, vous pouvez naviguer dans le rèsultat interactivement de droite á gauche, ce qui èvite que les lignes les plus longues soient coupèes. Cela rend la lecture du texte bien plus facile. Vous pouvez passer d'un mode á l'autre depuis la ligne de commande. Voyez l'aide pour plus de dètails sur less.
- Vous pouvez combiner des mèthodes trés complexes pour gèrer les rèsultats, comme, la possibilitè d'envoyer les rèsultats vers deux fichiers dans deux dossiers diffèrents, ou deux disques diffèrents, tout en affichant les rèsultats á l'ècran avec less :

  ```
  mysql> pager cat | tee /dr1/tmp/res.txt | \
  tee /dr2/tmp/res2.txt | less -n -i -S
  ```

Vous pouvez aussi combiner les deux fonctions ci-dessus : activer le `tee`, spècifier le `pager` 'less' et vous serez capable de naviguer dans les rèsultats avec le less Unix, tout en enregistrant tous les rèsultats dans un fichier. La diffèrence entre le `tee` d'Unix utilisè avec le `pager` et le `tee` intègrè du client `mysql`, est que le `tee` intègrè fonctionne même si vous n'avez pas de `tee` Unix disponible. Le `tee` enregistre tout ce qui est affichè á l'ècran, alors que le `tee` Unix utilisè avec `pager` n'en note pas autant. Enfin, le `tee` interactif est plus facile á activer et dèsactiver, lorsque vous souhaitez enregistrer un rèsultat dans un fichier, mais que vous voulez dèsactiver cette fonctionnalitè á d'autres moments.

Depuis MySQL version 4.0.2, il est possible de modifier l'invite de commande de `mysql`.

Vous pouvez utiliser les options de prompt suivantes :

Option	Description
\v	version de mysqld
\d	database en cours
\h	hôte MySQL
\p	port de connexion
\u	nom d'utilisateur
\U	Identifiant complet username@host
\\	'\'
\n	nouvelle ligne
\t	tabulation
\	espace
_	espace
\R	heure 24h (0-23)
\r	heure 12h (1-12)
\m	minutes
\y	année sur deux chiffres
\Y	année sur quatre chiffres
\D	format de date complet
\s	secondes
\w	jour de la semaine en trois lettres (Mon, Tue, ...)
\P	am/pm
\o	mois au format numèrique
\O	mois en trois lettres (Jan, Feb, ...)
\c	compteur du nombre de commande

'\' suivi de n'importe quelle lettre reprèsente la lettre littèralement.

Vous pouvez modifier l'invite de commande comme ceci :

Environment Variable

Vous pouvez utiliser la variable d'environnement `MYSQL_PS1`, en lui donnant la chaîne d'invite. Par exemple :

```
shell> export MYSQL_PS1="(\u@\h) [\d]> "
```

'my.cnf'

'.my.cnf' Vous pouvez configurer l'invite de commandes dans le fichier d'options MySQL. dans le groupe `mysql`. Par exemple :

```
[mysql]
prompt=(\u@\h) [\d]>\_
```

Command Line

Vous pouvez utiliser l'option de dèmarrage `--prompt` en ligne de commande avec `mysql`. Par exemple :

```
shell> mysql --prompt="(\u@\h) [\d]> "

(user@host) [database]>
```

Interactively

Vous pouvez aussi utiliser la commande `prompt` (ou `\R`) depuis le client pour modifier interactivement l'invite de commande. Par exemple :

```
mysql> prompt (\u@\h) [\d]>\_
PROMPT set to '(\u@\h) [\d]>\_'
(user@host) [database]>
(user@host) [database]> prompt
Returning to default PROMPT of mysql>
mysql>
```

4.8.3 `mysqladmin`, administrer un serveur MySQL

Un utilitaire pour exècuter des commandes d'administration. La syntaxe est :

```
shell> mysqladmin [OPTIONS] command [command-option] command ...
```

Vous pouvez obtenir une liste des options supportèes par votre version de `mysqladmin` avec la commande `mysqladmin --help`.

Le `mysqladmin` actuel supporte les commandes suivantes :

`create databasename`
: Crèe une nouvelle base.

`drop databasename`
: Efface une base et toutes ces tables.

`extended-status`
: Affiche un message de statut du serveur trés complet.

`flush-hosts`
: Vide tous les hôtes mis en cache.

`flush-logs`
: Vide de la mèmoire tous les logs.

`flush-tables`
: Vide de la mèmoire toutes les tables.

`flush-privileges`
: Recharger les tables de droits (identique á la commande reload).

`kill id,id,...`
: Termine un thread MySQL.

`password` Spècifie un nouveau mot de passe. Modifie l'ancien mot de passe en un nouveau.

`ping` Vèrifie si mysqld fonctionne ou pas.

`processlist`
: Affiche la liste des processus du serveur.

`reload` Recharge les tables de droits.

`refresh` Vide de la mèmoire toutes les tables, puis ferme et rèouvre les fichiers de logs.

`shutdown` Eteind le serveur.

`slave-start`
Dèmarre l'esclave de rèplication.

`slave-stop`
Eteind l'esclave de rèplication.

`status` Affiche le message de statut court du serveur.

`variables`
Affiche les variable disponibles.

`version` Affiche la version du serveur.

Toutes les commandes peuvent être rèduites á leur prèfixe simple. Par exemple :

```
shell> mysqladmin proc stat
+----+-------+-----------+----+-------------+------+-------+------+
| Id | User  | Host      | db | Command     | Time | State | Info |
+----+-------+-----------+----+-------------+------+-------+------+
| 6  | monty | localhost |    | Processlist | 0    |       |      |
+----+-------+-----------+----+-------------+------+-------+------+
Uptime: 10077  Threads: 1  Questions: 9  Slow queries: 0
Opens: 6 Flush tables: 1  Open tables: 2
Memory in use: 1092K  Max memory used: 1116K
```

La commande `mysqladmin status` liste les colonnes suivantes :

Colonne	**Description**
Uptime	Nombre de secondes de vie du serveur MySQL.
Threads	Nombre de threads actifs (clients).
Questions	Nombre de questions reçu des clients depuis le dèmarrage de `mysqld`.
Slow queries	Nombre de requêtes qui ont pris plus de `long_query_time` secondse. Voir Section 4.9.5 [Slow query log], page 331.
Opens	Combien de tables sont ouvertes par `mysqld`.
Flush tables	Nombre de commandes `flush ...`, `refresh` et `reload`.
Open tables	Nombre de tables qui sont ouvertes actuellement.
Memory in use	Mèmoire allouèe directement par `mysqld` (uniquement disponible si MySQL a ètè compilè avec l'option –with-debug=full).
Max memory used	Maximum de mèmoire allouèe directement par `mysqld` (uniquement disponible si MySQL a ètè compilè avec l'option –with-debug=full).

Si vous exècutez une commande `mysqladmin shutdown` sur une socket (en d'autres termes, sur un serveur oú `mysqld` fonctionne), `mysqladmin` va attendre que le fichier `pid-file` de MySQL soit effacè pour s'assurer que le serveur `mysqld` a ètè correctement stoppè.

4.8.4 Utiliser `mysqlcheck` pour l'entretien et la rèparation

Depuis MySQL version 3.23.38, vous pouvez utiliser un nouvel outil d'entretien et de rèparation pour les tables `MyISAM`. La diffèrence avec `myisamchk` est que `mysqlcheck` doit

être utilisè lorsque le serveur `mysqld` fonctionne, alors que `myisamchk` doit être utilisè lorsque le serveur ne fonctionne pas. L'intèrêt st que vous n'avez plus besoin d'interrompre le serveur pour vèrifier ou rèparer vos tables.

`mysqlcheck` utilise les commandes du serveur MySQL `CHECK`, `REPAIR`, `ANALYZE` et `OPTIMIZE`, d'une maniére pratique pour l'utilisateur.

Il y a trois façons diffèrentes d'utiliser `mysqlcheck`:

```
shell> mysqlcheck [OPTIONS] database [tables]
shell> mysqlcheck [OPTIONS] --databases DB1 [DB2 DB3...]
shell> mysqlcheck [OPTIONS] --all-databases
```

Il peut aussi être utilisè comme `mysqldump` lorsqu'il faut choisir les bases et tables á traiter.

`mysqlcheck` dispose d'une fonctionnalitè spèciale, comparè aux autres clients : le comportement par dèfaut, c'est á dire la vèrification des tables, peut être modifièe en renommant le fichier binaire. Si vous voulez avoir un fichier qui rèparer les tables par dèfaut, il suffit de copier `mysqlcheck` sur votre disque, et de l'appeler `mysqlrepair`, ou bien, de faire un lien symbolique sur l'exècutable et de l'appeler `mysqlrepair`. SI vous appelez `mysqlrepair`, il va rèparer les tables par dèfaut.

Les noms que vous pouvez utiliser pour que `mysqlcheck` rèparer automatiquement les tables sont :

```
mysqlrepair:   L'option par dèfaut est -r
mysqlanalyze:  L'option par dèfaut est -a
mysqloptimize: L'option par dèfaut est -o
```

Les options disponibles pour `mysqlcheck` sont listèes ici. Vèrifiez que votre version les supporte avec la commande `mysqlcheck --help`.

`-A, --all-databases`
: Vèrifie toutes les bases. C'est la même chose que –databases dans toutes les bases sèlectionnèes.

`-1, --all-in-1`
: Au lieu de faire une requête par table, exècute toutes les requêtes dans une requête, sèparèment pour chaque base. Les noms de tables seront sèparès par une virgule.

`-a, --analyze`
: Analyse les tables indiquèes.

`--auto-repair`
: Si une table vèrifièes est corrompue, la corrige automatiquement. La rèparation sera faite aprés la vèrification de toutes les tables, si des tables corrompues ont ètè dècouvertes.

`-#, --debug=...`
: Affiche le log de debug. Souvent, c'est dans 'd:t:o,filename'

`--character-sets-dir=...`
: Dossier contenant le jeu de caractéres

`-c, --check`
: Vèrifie les tables en erreur

`-C, --check-only-changed`
: Vèrifie uniquement les tables qui ont ètè modifièes depuis la derniére modification, ou qui n'ont pas ètè correctement fermèes.

`--compress`
: Utiliser la compression du protocole client/serveur.

`-?, --help`
: Affiche ce message d'aide, et termine.

`-B, --databases`
: Pour tester plusieurs bases de donnèes. Notez que la diffèrence d'utilisation : dans ce cas, aucune table n'est prècisè. Tous les arguments de noms sont considèrès comme des noms de base.

`--default-character-set=...`
: Spècifie le jeu de caractéres par dèfaut.

`-F, --fast`
: Ne vèrifie que les tables qui n'ont pas ètè correctement fermèes.

`-f, --force`
: Continue même si on rencontre une erreur SQL.

`-e, --extended`
: Si vous utilisez cette option avec CHECK TABLE, elle va s'assurer que la table est totalement cohèrente, mais prendre un trés long temps.

  Si vous utilisez cette option avec REPAIR TABLE, elle va rèaliser une rèparation exhaustive de la table, qui peut non seulement prendre un temps trés long, mais produire de nombreuses lignes erronèes.

`-h, --host=...`
: Connexion á l'hôte.

`-m, --medium-check`
: Plus rapide que la vèrification compléte, mais ne trouvera que 99.99 % de toutes les erreurs. Cela devrait être la bonne option pour la plupart des situations.

`-o, --optimize`
: Optimise la table

`-p, --password[=...]`
: Le mot de passe á utiliser lors de la connexion au serveur. Si aucun mot de passe n'est fourni, il sera demandè en ligne de commande.

`-P, --port=...`
: Le numèro de port de la connexion.

`-q, --quick`
: Si vous utilisez cette option avec CHECK TABLE, elle va èviter que l'analyse ne scanne les lignes pour vèrifier les mauvais liens. C'est la mèthode d'analyse la plus rapide.

  Si vous utilisez cette option avec REPAIR TABLE, elle va essayer de ne rèparer que le fichier d'index. C'est la mèthode la plus rapide pour la rèparation.

`-r, --repair`
Peut corriger presque tout, sauf les problémes de doublons pour les clès uniques.

`-s, --silent`
Affiche moins de messages d'erreurs.

`-S, --socket=...`
Nom du fichier de socket á utiliser pour la connexion.

`--tables` Remplace l'option –databases (-B).

`-u, --user=#`
Nom d'utilisateur pour la connexion, si ce n'est pas l'utilisateur courant.

`-v, --verbose`
Afficher des informations sur les diffèrentes ètapes.

`-V, --version`
Affiche les informations de version, et termine.

4.8.5 `mysqldump`, exporter les structures de tables et les donnèes

Utilitaire qui permet d'exporter une base ou un groupe de bases vers un fichier texte, pour la sauvegarde ou le transfert entre deux serveurs (pas nècessairement entre serveurs MySQL). L'export contiendra les requêtes SQL nècessaires pour crèer la table et la remplir.

Si vous faîtes une sauvegarde du serveur, vous devriez aussi utiliser la commande `mysqlhotcopy`. Voir Section 4.8.6 [`mysqlhotcopy`], page 322.

```
shell> mysqldump [OPTIONS] database [tables]
OR     mysqldump [OPTIONS] --databases [OPTIONS] DB1 [DB2 DB3...]
OR     mysqldump [OPTIONS] --all-databases [OPTIONS]
```

Si vous ne spècifiez pas de table, ou si vous utilisez l'option `--databases` ou `--all-databases`, la base de donnèes compléte sera exportèe.

Vous pouvez obtenir une liste des options valides pour votre version de `mysqldump` avec la commande `mysqldump --help`.

Notez que si vous exècutez `mysqldump` sans l'option `--quick` ou `--opt`, `mysqldump` va charger la totalitè du rèsultat en mèmoire, avant de l'ècrire. Cette option peut rèsoudre des problémes de mèmoire si vous exportez de grosses tables.

Notez que si vous utilisez une nouvelles copie du programme `mysqldump`, et que vous allez faire un export qui sera lu dans une vieille version de MySQL, vous ne devriez pas utiliser les options `--opt` et `-e`.

`mysqldump` supporte les options suivantes :

`--add-locks`
Ajoute une commande `LOCK TABLES` avant l'export de table et une commande `UNLOCK TABLE` aprés(Pour accèlèrer les inserionts dans MySQL).

`--add-drop-table`
Ajoute une commande `drop table` avant chaque requête de crèation de table.

`-A, --all-databases`
: Exporte toutes les tables. C'est l'èquivalent de l'option `--databases` avec toutes les bases de donnèes sèlectionnèes.

`-a, --all` Inclut toutes les options de crèations de table spècifiques á MySQL.

`--allow-keywords`
: Permet la crèation de colonnes ayant des noms de mots rèservès. Cela fonctionne en prèfixant chaque nom de colonne avec le nom de la table.

`-c, --complete-insert`
: Utilise une commande compléte d'insertion (avec les noms des colonnes).

`-C, --compress`
: Compresse toutes les informatiosn entre le client et le serveur, les deux supporte la compression.

`-B, --databases`
: Pour exporter plusieurs bases de donnèes. Notez la diffèrence d'utilisation. Dans ce cas, aucune table n'est spècifiè. Tous les arguments de noms sont considèrès comme des noms de base. Une ligne `USE db_name;` sera ajoutèe dans l'export avant chaque base de donnèes.

`--delayed`
: Les insertions se font avec la commande `INSERT DELAYED`.

`-e, --extended-insert`
: Utilise la nouvelle syntaxe multi-ligne `INSERT`. (Cela donne des insertions plus courtes et plus efficaces).

`-#, --debug[=option_string]`
: Trace l'utilisation du programme (pour le dèbogage).

`--help` Affiche le message d'aide et quitte.

`--fields-terminated-by=...`
`--fields-enclosed-by=...`
`--fields-optionally-enclosed-by=...`
`--fields-escaped-by=...`
`--lines-terminated-by=...`
: Ces options sont utilisèes avec l'option `-T` et ont la même signification que les clauses correspondantes de la commande `LOAD DATA INFILE`. Voir Section 6.4.9 [LOAD DATA], page 497.

`-F, --flush-logs`
: crit tout le fichier de log du serveur avant de commencer l'export.

`-f, --force,`
: Continue même si une erreur SQL survient durant l'export.

`-h, --host=..`
: Exporte les donnèes depuis le serveur MySQL vers l'hôte indiquè. L'hôte par dèfaut est `localhost`.

`-l, --lock-tables.`
: Verrouille toutes les tables avant de commencer l'export. Les tables sont verrouillèes avec `READ LOCAL` pour permettre des insertions concurentes sur les tables `MyISAM`.

  Notez que lorsque vous exportes des tables de bases diffèrentes, l'option `--lock-tables` va verrouiller chaque base sèparèment. Cette option ne vous garantira pas que vos tables seront logiquement cohèrente entre les bases. Des tables de diffèrentes bases pourraient être exportèes dans des ètats trés diffèrents.

`-K, --disable-keys`
: `/*!40000 ALTER TABLE tb_name DISABLE KEYS */;` et `/*!40000 ALTER TABLE tb_name ENABLE KEYS */;` seront ajoutès dans le rèsultat. Cela rendra les chargements de donnèes plus rapides sur les serveurs MySQL 4.0 car les index sont alors crèès aprés l'insertion de toutes les donnèes.

`-n, --no-create-db`
: `CREATE DATABASE /*!32312 IF NOT EXISTS*/ db_name;` ne sera pas ajoutè dans l'export. Sinon, la ligne ci-dessus sera ajoutèe, si l'une des options `--databases` ou `--all-databases` ont ètè activèe.

`-t, --no-create-info`
: N'ècrit pas les informations de crèation de table (la requête `CREATE TABLE`).

`-d, --no-data`
: N'ècrit aucune ligne d'informations sur la atble. C'est trés pratique si vous voulez simplement exporter la structure de la table.

`--opt`
: Identique á `--quick --add-drop-table --add-locks --extended-insert --lock-tables`. Vous obtiendrez l'export le plus rapide á importer dans un serveur MySQL.

`-pyour_pass, --password[=your_pass]`
: Le mot de passe á utiliser lors de la connexion au serveur. Si vous spècifiez en omettant la partie '`=your_pass`', `mysqldump` vous demandera le mot de passe en ligne de commande.

`-P port_num, --port=port_num`
: Le port TCP/IP á utiliser avec l'hôte. Ceci sert pour les connexions d'hôte á hôte, autre que `localhost`, pour lequel les sockets Unix seront utilisèes).

`-q, --quick`
: Ne garde pas en buffer les requêtes, mais ècrit immèdiatement dans la sortie. Utilise `mysql_use_result()` pour cela.

`-Q, --quote-names`
: Protége les noms des tables et colonnes avec le caractére ‘‘.

`-r, --result-file=...`
: Ecrit directemetn dans le fichier indiquè. Cette option doit être utilisè sur MSDOS, car cela èvite que la nouvelle ligne '`\n`' soient converties en '`\n\r`' (nouvelle ligne et retour chariot).

`--single-transaction`
: Cette option ajoute la commande SQL `BEGIN` avant d'exporter les donnèes vers le serveur. C'est gènèralement pratique pour les tables `InnoDB` et le niveau d'isolation de transaction `READ_COMMITTED`, car ce mode va exporter l'ètat de la base au moment de la commande `BEGIN` sans bloquer les autres applications.

  Lorsque vous utilisez cette option, pensez bien que seules les tables transactionnelles seront exportèes dans un ètat cohèrent, c'est á dire que les tables `MyISAM` ou `HEAP` qui seront exportèes avec cette option, pourront changer d'ètat.

  L'option `--single-transaction` a ètè ajoutèe en version 4.0.2. Cette option est mutuellement exclusive avec l'option `--lock-tables` car `LOCK TABLES` va valider une transaction interne prècèdente.

`-S /path/to/socket, --socket=/path/to/socket`
: Le fichier de socket á utiliser lors de la connexion á `localhost` (qui est l'hôte par dèfaut).

`--tables` Remplace l'option –databases (-B).

`-T, --tab=path-to-some-directory`
: Crèe un fichier `table_name.sql`, qui contient les commandes SQL CREATE, et un fichier `table_name.txt`, qui contient les donnèes, pour chaque table. Le format du fichier '`.txt`' est celui qui est spècifiè par les options `--fields-xxx` et `--lines--xxx`. **Note** : cette option ne fonctionne qui si `mysqldump` est exècutè sur la même machine que le dèmon `mysqld`, et que le nom d'utilisateur et le groupe de `mysqld` (normalement l'utilisateur `mysql`, et le groupe `mysql`) doivent avoir des permission pour crèer et ècrire un fichier dans le dossier que vous spècifiez.

`-u user_name, --user=user_name`
: Le nom d'utilisateur MySQL lors de la connexion á un serveur distant. La valeur par dèfaut est votre nom d'utilisateur Unix.

`-O var=option, --set-variable var=option`
: Spècifie la valeur d'une variable. Les noms possibles des variables sont spècifiès ci-dessous. Notez bien que `--set-variable` est obsoléte depuis MySQL 4.0, il vous suffit alors d'utiliser la syntaxe `--var=option`.

`-v, --verbose`
: Mode dètaillè. Affiche plus d'informations sur les faits et gestes du programme.

`-V, --version`
: Affiche la version du programme et quitte.

`-w, --where='where-condition'`
: Exporte uniquement les lignes sèlectionnèes. Notez que les guillemets sont obligatoires.

`-X, --xml` Exporte la base au format XML.

`-x, --first-slave`
: Verrouille toutes les tables dans les bases.

```
"--where=user='jimf'" "-wuserid>1" "-wuserid<1"
```

`-O net_buffer_length=#, where # < 16M`
: Lors de la crèation de commandes d'insertions multilignes, (comme avec l'option `--extended-insert` ou `--opt`), `mysqldump` va crèer des lignes jusqu'á la taille de `net_buffer_length`. Si vous augmentez cette valeur, vous devriez aussi vous assurer que la variable MySQL `max_allowed_packet` est plus grande que `net_buffer_length`.

L'usage normal de `mysqldump` est probablement de faire des sauvegardes de bases. Voir Section 4.4.1 [Backup], page 242.

```
mysqldump --opt database > backup-file.sql
```

Vous pouvez importer les données dans la base MySQL avec :

```
mysql database < backup-file.sql
```

ou

```
mysql -e "source /patch-to-backup/backup-file.sql" database
```

Cependant, il est trés pratique pour remplir un autre serveur MySQL avec des informations depuis une base :

```
mysqldump --opt database | mysql ---host=remote-host -C database
```

Il est possible d'exporter plusieurs bases de données en une seule commande :

```
mysqldump --databases database1 [database2 ...] > my_databases.sql
```

Si vous souhaitez exporter toutes les bases, vous pouvez utiliser :

```
mysqldump --all-databases > all_databases.sql
```

4.8.6 `mysqlhotcopy`, copier les bases et tables MySQL

`mysqlhotcopy` est un script Perl qui utilise `LOCK TABLES`, `FLUSH TABLES` et `cp` ou `scp` pour faire rapidement des sauvegardes de bases. C'est la mèthode la plus rapide pour faire une sauvegarde de bas. C'est aussi le moyen le plus sûr pour copier des tables et bases, mais il ne peut fonctionner que sur la machine qui contient les fichiers de données.

```
mysqlhotcopy db_name [/path/to/new_directory]

mysqlhotcopy db_name_1 ... db_name_n /path/to/new_directory

mysqlhotcopy db_name./regex/
```

`mysqlhotcopy` supporte les options suivantes :

`-?, --help`
: Affiche un ècran d'aide et quitte.

`-u, --user=#`
: Nom d'utilisateur pour la connexion au serveur.

`-p, --password=#`
: Mot de passe utilisè pour la connexion au serveur.

`-P, --port=#`
: Port utilisè pour la connexion au serveur.

`-S, --socket=#`
: Socket utilisè pour la connexion au serveur.

`--allowold`
: Ne pas annuler si la sauvegarde existe dèjá (renomme la simplement en _old)

`--keepold`
: Ne pas effacer une sauvegarde prècèdente (celle qui a ètè renommèe) lorsque c'est terminè.

`--noindices`
: Ne pas inclure les fichiers d'index complet dans la copie, pour faire des fichiers de sauvegarde plus petit et plus rapide. Les index peuvent toujours être reconstruits plus tard avec `myisamchk -rq.`.

`--method=#`
: Mèthode de copie (`cp` ou `scp`).

`-q, --quiet`
: Mode silencieux. N'affiche que les erreurs.

`--debug` Active le dèbogage.

`-n, --dryrun`
: Rapporte les actions rèalisèes sans les faire.

`--regexp=#`
: Copie toutes les bases dont le nom vèrifie un masque d'expression règuliére.

`--suffix=#`
: Suffixe des noms des bases copièes.

`--checkpoint=#`
: Insére un point de contrôle dans la table spècifièe (base.table)

`--flushlog`
: Vide les logs sur le disque une fois que toutes les tables sont verrouillèes.

`--tmpdir=#`
: Dossier temporaire (au lieu de /tmp).

Vous pouvez essayer `perldoc mysqlhotcopy` pour avoir plus de documentation sur `mysqlhotcopy`.

`mysqlhotcopy` lit les options du groupe `[client]` et `[mysqlhotcopy]` dans le fichier d'options.

Pour être capable d'exècuter `mysqlhotcopy`, vous avez besoin des droits d'ècriture dans le dossier de sauvegarde, et le droit de `SELECT` dans les tables que vous souhaitez copier, ainsi que les droits MySQL de `RELOAD` (pour utiliser la commande `FLUSH TABLES`).

4.8.7 `mysqlimport`, importer des donnèes depuis des fichiers texte

`mysqlimport` fournit une interface en ligne de commande á la commande SQL `LOAD DATA INFILE`. La plupart des options de `mysqlimport` correspondent directement aux mêmes options de `LOAD DATA INFILE`. Voir Section 6.4.9 [`LOAD DATA`], page 497.

`mysqlimport` est appelè comme ceci :

```
shell> mysqlimport [options] database textfile1 [textfile2 ...]
```

Pour chaque fichier texte indiquè dans la ligne de commande, `mysqlimport` supprime toute extension du nom du fichier, et utilise le rèsultat pour dèterminer le nom de la table qui va recevoir le contenu du fichier. Par exemple, pour des fichiers appelès '`patient.txt`', '`patient.text`' et '`patient`' seront tous importès dans la table `patient`.

`mysqlimport` supporte les options suivantes :

`-c, --columns=...`
: Cette option prend une liste de noms de colonnes, sèparès par des virgules. Ce champs est utilisè pour crèer une commande `LOAD DATA INFILE` correcte, qui sera alors passèe á MySQL. Voir Section 6.4.9 [`LOAD DATA`], page 497.

`-C, --compress`
: Compresse toutes les informations entre le client et le serveur, si c'est possible.

`-#, --debug[=option_string]`
: Trace l'utilisation du programme (pour le dèbogage).

`-d, --delete`
: Vide la table avant d'importer le fichier texte.

`--fields-terminated-by=...`
`--fields-enclosed-by=...`
`--fields-optionally-enclosed-by=...`
`--fields-escaped-by=...`
`--lines-terminated-by=...`
: Ces options ont la même signification que les clauses correspondantes de `LOAD DATA INFILE`. Voir Section 6.4.9 [`LOAD DATA`], page 497.

`-f, --force`
: Ignore les erreurs. Par exemple, si une table n'existe pas pour un fichier texte, `mysqlimport` va continuer de traiter les autres fichiers. Sans `--force`, `mysqlimport` se termine dés qu'une erreur survient.

`--help`
: Affiche le message d'aide et quitte.

`-h host_name, --host=host_name`
: Importe les donnèes sur le serveur MySQL, avec l'hôte spècifiè. La valeur par dèfaut est `localhost`.

`-i, --ignore`
: Voir la description de `--replace`.

`-l, --lock-tables`
: Verrouille **toutes** les tables en ècriture avant de ne traiter les fichiers textes. Cela assure que toutes les tables sont synchronisèe sur le serveur.

`-L, --local`
: Lit le fichier d'entrèe dans le client. Par dèfaut, les fichiers textes sont supposès être lus par le serveur, si vous vous connectez á `localhost` (qui l'hôte par dèfaut).

`-pyour_pass, --password[=your_pass]`
: Le mot de passe á utiliser lors de la connexion au serveur. Si vous ne spècifiez pas la partie '`=your_pass`', `mysqlimport` va vous demander le mot de passe en ligne.

`-P port_num, --port=port_num`
: Le port TCP/IP utilisè avec l'hôte. Cela sert pour les connexions á des hôtes qui ne sont pas `localhost`, pour lequel la socket Unix est utilisèe.

`-r, --replace`
: Les options `--replace` et `--ignore` contrôles la gestion des lignes lues envers les lignes qui existent dèjá sur le serveur. Si vous spècifiez l'option `--replace`, les nouvelles lignes remplaceront les lignes existantes. Si vous spècifiez `--ignore`, les lignes qui sont en double dans une table qui dispose d'une colonne de type unique. Si vous ne spècifiez pas ces options, une erreur surviendra lorsqu'une clè en double sera trouvèe, et la lecture du reste du fichier sera annulè.

`-s, --silent`
: Mode silencieux. N'affiche que les erreurs qui surviennent.

`-S /path/to/socket, --socket=/path/to/socket`
: Le fichier de socket á utiliser lors de la connexion á `localhost` (qui est l'hôte par dèfaut).

`-u user_name, --user=user_name`
: Le nom de l'utilisateur MySQL á utiliser lors de la connexion au serveur MySQL. La valeur par dèfaut est celui de votre utilisateur Unix.

`-v, --verbose`
: Mode dètaillè. Affiche bien plus d'informations sur les actions du programme.

`-V, --version`
: Affiche la version et quitte.

Voici un exemple d'utilisation de `mysqlimport` :

```
$ mysql --version
mysql  Ver 9.33 Distrib 3.22.25, for pc-linux-gnu (i686)
$ uname -a
Linux xxx.com 2.2.5-15 #1 Mon Apr 19 22:21:09 EDT 1999 i586 unknown
$ mysql -e 'CREATE TABLE imptest(id INT, n VARCHAR(30))' test
$ ed
a
100     Max Sydow
101     Count Dracula
.
w imptest.txt
32
q
$ od -c imptest.txt
0000000   1   0   0  \t   M   a   x       S   y   d   o   w  \n   1   0
0000020   1  \t   C   o   u   n   t       D   r   a   c   u   l   a  \n
```

```
0000040
$ mysqlimport --local test imptest.txt
test.imptest: Records: 2  Deleted: 0  Skipped: 0  Warnings: 0
$ mysql -e 'SELECT * FROM imptest' test
+------+---------------+
| id   | n             |
+------+---------------+
|  100 | Max Sydow     |
|  101 | Count Dracula |
+------+---------------+
```

4.8.8 Afficher les bases, tables et colonnes

`mysqlshow` peut être utilisè pour lister les bases qui existent, leurs tables et leurs colonnes.

Avec le programme `mysql` vous pouvez obtenir des informations avec la commande `SHOW`. Voir Section 4.5.6 [SHOW], page 267.

`mysqlshow` est utilisè comme ceci :

```
shell> mysqlshow [OPTIONS] [database [table [column]]]
```

- Si aucune base n'est indiquèe, toutes les bases de donnèes sont listèes.
- Si aucune table n'est nommèe, toutes les tables de la base sont affichèes.
- Si aucune colonne n'est nommèe, toutes les colonnees et leur type sont affichès.

Notez que dans les nouvelles versions de MySQL, vous ne verrez que les bases de donnèes, tables et colonnes pour lesquelles vous avez des droits.

Si le dernier argument contient un caractére joker shell ou SQL (`*`, `?`, `%` ou `_`) alors seules les entitès qui valident ce masque sont affichèes. Si une base contient des caractéres soulignès, ils doivent être protègès avec un anti-slash (certains shell Unix en demande même deux), afin de lister correctement les tables et les colonnes. Les '*' sont convertis en SQL '%' et '?' en SQL '_'. Cela peut causer des confusions lorsque vous essayez d'afficher des colonnes qui contiennent un soulignè `_`, comme c'est le cas avec `mysqlshow` qui ne vous affiche que les colonnes qui vèrifient le masque. Ceci est facilement corrigè en ajoutant un caractére `%` en plus dans la ligne de commande (comme argument sèparè).

4.8.9 `perror`, expliquer les codes d'erreurs

Pour la plupart des erreurs systéme, MySQL va, en plus d'un message interne, aussi afficher un code d'erreur, dans l'un des styles suivants : `message ... (errno: #)` or `message ... (Errcode: #)`.

Vous pouvez dècouvrir ce que ce code d'erreur signifie soit en examinant la documentation de votre systéme, soit en utilisant l'utilitaire `perror`.

`perror` affiche une description pour le code d'erreur, ou, pour une erreur du gestionnaire de tables MyISAM/ISAM.

`perror` est appelè comme ceci :

```
shell> perror [OPTIONS] [ERRORCODE [ERRORCODE...]]

Example:

shell> perror 13 64
Error code  13:  Permission denied
Error code  64:  Machine is not on the network
```

Notez que les messages d'erreurs sont dèpendants du systéme.

4.8.10 Comment exècuter des commandes SQL depuis un fichier texte

Le client `mysql` peut être utilisè en mode interactif comme ceci :

```
shell> mysql database
```

Toutefois, il est aussi possible de rassembler les commandes SQL dans un fichier, et d'indiquer á `mysql` de lire les commandes dans ce fichier. Pour faire cela, crèez un fichier texte '`fichier_texte`' qui contient les commandes SQL que vous souhaitez exècuter. Puis, exècutez ce fichier avec `mysql` comme ceci :

```
shell> mysql database < fichier_texte
```

Vous pouvez aussi dèmarrer votre fichier texte avec la commande `USE nom_base`. Dans ce cas, il n'est pas nèeccaire de spècifier le nom de la base de donnèes dans la ligne de commande :

```
shell> mysql < fichier_texte
```

Si vous avez dèjá dèmarrè le client `mysql`, vous pouvez exècuter un script SQL en utilisant la commande `source` :

```
mysql> source nom_fichier;
```

Pour plus d'informations sur le mode batch, consultez Section 3.6 [Batch mode], page 186.

4.9 Les fichiers de log de MySQL

MySQL a plusieurs fichiers de log qui peuvent vous aider á savoir ce qui se passe á l'intèrieur de `mysqld`:

Fichier	Description
Le log d'erreurs	Problémes rencontrès lors du dèmarrage, de l'exècution ou du stoppage de `mysqld`.
Le log isam	Garde une trace des changements liès au tables ISAM. Utilisè uniquement pour dèboguer le code isam.
Le log de requêtes	Connexions ètablies et requêtes exècutèes.
Le log de mises á jour	Dèsapprouvè : Enregistre toutes les commandes qui changent les donnèes.
Le log binaire	Enregistre toutes les commandes qui changent quelque chose. Utilisè pour la replication.
Le log des requêtes lentes	Enregistre toutes les requêtes qui ont pris plus de `long_query_time` á s'exècuter ou celles qui n'ont pas utilisè d'index.

Tous les fichiers de log peuvent être trouvès dans le dossier de donnèes de `mysqld`. Vous pouvez forcer `mysqld` á rèouvrir les fichiers de log (ou dans quelques cas á passer á un nouveau log) en exècutant `FLUSH LOGS`. Voir Section 4.5.3 [FLUSH], page 265.

4.9.1 Le log d'erreurs

`mysqld` affiche toutes les erreurs dans stderr, le script `safe_mysqld` les redirige vers un fichier nommè `'hostname'.err`. (Sur Windows, `mysqld` les ècrit directement dans '`\mysql\data\mysql.err`'.)

Ce fichier contient des informations indiquant quand `mysqld` a ètè dèmarrè et stoppè pour cause d'erreurs critiques rencontrè lors de son fonctionnement. Si `mysqld` se termine inopinèment et que `safe_mysqld` a besoin de redèmarrer `mysqld`, `safe_mysqld` ajoutera une ligne `restarted mysqld` dans ce fichier. Ce log contient aussi un avertissement si `mysqld` trouve une table qui a besoin d'être automatiquement vèrifièe ou rèparèe.

Sur quelques systémes d'exploitation, le log d'erreurs contiendra un traçage de la pile mèmoire relatif á l'endroit oú `mysqld` s'est terminè. Il peut être utilisè pour trouver les raisons du crash de `mysqld`. Voir Section E.1.4 [Using stack trace], page 818.

4.9.2 Le log gènèral de requêtes

Si vous voulez savoir ce qui se passe á l'intèrieur de `mysqld`, vous devez le dèmarrer avec `--log[=fichier]`. Cela aura pour effet d'ècrire toutes les connexions et les requêtes dans le fichier de log (pas dèfaut nommè '`'hostname'.log`'). Ce log peut être trés utile quand vous suspectez une erreur dans un client et voulez savoir exactement ce que `mysqld` pense que le client lui a envoyè.

Les anciennes versions du script `mysql.server` (de MySQL 3.23.4 á 3.23.8) passent á `safe_mysqld` une option `--log` (active le log gènèral de requêtes). Si vous avez besoin de meilleurs performances lorsque vous dèmarrez MySQL dans un environnement de production, vous pouvez supprimer l'option `--log` de `mysql.server` ou la changer en `--log-bin`. Voir Section 4.9.4 [Binary log], page 329.

Les entrèes dans ce log sont ècrites en même temps que `mysqld` reçoit une question. Cela peut être diffèrent de l'ordre dans lequel les commandes sont exècutèes. Cela est contraire au log des mises á jour et du log binaire qui sont ècrits aprés l'exècution de la requête, mais avant les verrouillages.

4.9.3 Le log de modification

Note : le log de modifications a ètè remplacè par le log binaire. Voir Section 4.9.4 [Binary log], page 329. Avec ce nouveau log, vous pouvez faire tout ce que vous faisiez avec le log de modifications.

Lors l'option `--log-update[=file_name]` est utilisèe au dèmarrage, `mysqld` ècrit un fichier de log contenant toutes les commandes SQL qui modifie les donnèes. Si aucun fichier n'est spècifiè, il prendra la valeur par dèfaut du nom de l'hôte. Si un fichier est spècifiè mais qu'aucun chemin n'est indiquè, le fichier sera ècrit dans le dossier de donnèes. Si le fichier '`file_name`' n'a pas d'extension, `mysqld` va crèer un fichier de log avec ce

nom : '`file_name.###`', oú `###` est un nombre qui s'incrèmente á chaque vois que vous exècutez la commande `mysqladmin refresh`, `mysqladmin flush-logs`, `FLUSH LOGS` ou que vous redèmarrez le serveur.

Note : pour que la technique ci-dessus fonctionne, vous ne devez pas crèer de fichiers avec le nom du fichier de log + une extension, qui pourrait être considèrèe comme un nombre, dans le dossier qui contient les log de modifications.

Si vous utilisez les options `--log` ou `-l`, `mysqld` ècrit un log gènèral avec le nom de '`hostname.log`', et les redèmarrage ou les rafraîchissements de fichiers de logs ne gènèreront pas de nouveaux fichiers (même si le fichier lui même est bien refermè, puis ouvert). Dans ce cas, vous pouvez le copier sous Unix avec :

```
mv hostname.log hostname-old.log
mysqladmin flush-logs
cp hostname-old.log to-backup-directory
rm hostname-old.log
```

Le fichier de log de modification est trés utile, car il n'enregistre que les commandes qui modifient les donnèes. Ce qui fait qu'une commande `UPDATE` ou `DELETE` avec une requête `WHERE` qui ne trouve pas de lignes ne seront pas ècrites dans ce fichier. Il va même ignorer les requêtes `UPDATE` qui modifie une colonne avec la valeur dèjá prèsente.

L'enregistrement dans le log de modification est fait juste aprés l'achévement de la requête, mais avant la levèe des verrous, et les validations. Cela garantit que la requête sera enregistrèe.

Si vous voulez modifier une base grâce aux fichier de log de modification, vous pouvez utiliser la commande suivante (en supposant que vos fichiers de log de modification porte le nom de '`file_name.###`') :

```
shell> ls -1 -t -r file_name.[0-9]* | xargs cat | mysql
```

`ls` est utilisè pour obtenir toute la liste des fichiers de logs du dossier.

Ceci peut être utile si vous devez repartir d'un fichier de sauvegarde aprés un crash, et que vous souhaitez rèexècuter les modifications qui ont eu lieu depuis la sauvegarde.

4.9.4 Le log binaire de modifications

Notre objectif est de remplacer l'utilisation du log de modifications par le log binaire de modifications. Nous vous recommandons donc de changer de format de log le plus tôt possible.

Le log binaire contient toutes les informations qui sont disponibles dans le log de modifications, dans un format bien plus efficace. Il contient aussi des informations sur le temps d'exècution de la requête. Il ne contient pas les requêtes qui n'ont pas modifiè les donnèes. Si vous voulez enregistrer toutes les requêtes, vous devez utiliser le fichier de log gènèral. Voir Section 4.9.2 [Query log], page 328.

Le log binaire est aussi utilisè lorsque de la rèplication d'un maître par un esclave. Voir Section 4.10 [Replication], page 332.

Lorsque l'option de dèmarrage `--log-bin[=file_name]` est utilisèe, `mysqld` ècrit un fichier de log contenant toutes les commandes SQL qui modifient les donnèes. Si aucun nom de

fichier n'est donnè, le nom de la machine hôte est utilisè, suivi de `-bin`. Si un nom est donnè, mais qu'il ne contient pas de chemin, le fichier sera ècrit dans le dossier de donnèes.

Si vous fournissez une extension á `--log-bin=filename.extension`, l'extension sera automatiquement supprimèe.

Au nom du fichier de log binaire, `mysqld` va ajouter une extension qui est un nombre automatiquement incrèmentè chaque fois que vous exècutez `mysqladmin refresh`, `mysqladmin flush-logs`, `FLUSH LOGS` ou redèmarrez le serveur. Un nouveau fichier de log sera automatiquement crèè lorsque le fichier en cours atteint la taille de `max_binlog_size`. Vous pouvez effacer tous les fichiers de log inactifs avec la commande `RESET MASTER`. Voir Section 4.5.4 [RESET], page 266.

Vous pouvez utiliser les options suivantes avec `mysqld` pour modifier ce qui est enregistrè dans le fichier de log :

Option	Description
`binlog-do-db=database_name`	Indique au maître qu'il doit enregistrer les modifications des bases spècifièes, et ignorer les autres. (Example: `binlog-do-db=some_database`)
`binlog-ignore-db=database_name`	Indique autre maître que les modifications des bases spècifièes ne doit pas être enregistrèes. (Exemple : `binlog-ignore-db=some_database`)

Pour être capable de connaître les diffèrents fichiers de logs qui ont ètè utilisès, `mysqld` va aussi crèer un fichier d'index qui contient les noms des fichiers de log utilisè. Par dèfaut, ils ont le même nom que le fichier de log, avec l'extension `'.index'`. Vous pouvez modifier le nom du fichier d'index avec l'option `--log-bin-index=[filename]`.

Si vous utilisez la rèplication, vous ne devez pas effacer les anciens log binaires jusqu'á ce que vous soyez sûrs que les esclaves n'en auront plus besoin. Une façon de faire cela est d'utiliser la commande `mysqladmin flush-logs` une fois par jour, et d'effacer les fichiers de log qui ont plus de trois jours.

Vous pouvez examiner le fichier de log binaire avec la commande `mysqlbinlog`. Par exemple, vous pouvez mettre á jour le serveur MySQL depuis la ligne de commande comme ceci :

```
shell> mysqlbinlog log-file | mysql -h server_name
```

Vous pouvez aussi utiliser le programme `mysqlbinlog` pour lire le fichier de log binaire directement dans le serveur MySQL.

`mysqlbinlog --help` vous donnera plus d'informations sur comment utiliser ce programme.

Si vous utilisez `BEGIN [WORK]` ou `SET AUTOCOMMIT=0`, vous devez utiliser le fichier de log binaire pour les sauvegardes, plutôt que le vieux fichier de log de modifications.

L'enregistrement dans le fichier de log binaire est fait immèdiatement aprés l'achévement de la requête, mais avant la libèration des verrous ou la validation de la requête. Cela garantit que les requêtes seront enregistrèes dans l'ordre d'exècution.

Les modifications dans les tables non transactionnelles sont enregistrèes dans le fichier de log binaire immèdiatement aprés exècution. Pour les tables transactionnelles comme `BDB` ou `InnoDB`, toutes les modifications (`UPDATE`, `DELETE` ou `INSERT`) qui modifient les tables sont mises en cache jusqu'á ce qu'une commande `COMMIT` ne les envoie au serveur. A ce

moment, `mysqld` ècrit la totalitè de la transaction dans le log binaire, avant d'appliquer la commande `COMMIT`. Tous les threads vont, au dèmarrage, allouer un buffer de la taille de `binlog_cache_size` octets pour enregistrer les requêtes. Si la requête est plus grande que ce buffer, le thread va ouvrir un fichier temporaire pour ècrire la transaction. Le fichier temporaire sera supprimè dés que le thread se termine.

L'option `max_binlog_cache_size` (par dèfaut 4Go) peut être utilisè pour limiter la taille utilisèe pour mettre en cache une transaction multi-requête. Si la transaction est plus grande que que cette taille, elle sera annulèe.

Si vous utilisez les log de modification ou binaire, les insertions concurrentes seront converties en insertions normales lors de l'utilisation de `CREATE ... SELECT` ou `INSERT ... SELECT`. Cela garantit que vous pourrez recrèer une copie exacte de la table en appliquant les même commandes sauvegardèes.

4.9.5 Le log des requêtes lentes

Lorsqu'il est dèmarrè avec l'option `--log-slow-queries[=file_name]`, `mysqld` va ècrire dans un fichier les requêtes SQL qui vont mettre plus de `long_query_time` secondes á s'exècuter. Le temps d'acquisition d'un verrou n'est pas comptè.

Les requêtes lentes sont enregistrèes aprés l'achévement de l'exècution de la requête, et libèration du verrou. Cela peut être diffèrent de l'ordre dans lequel les commandes sont exècutèes.

Si aucun nom de fichier n'est donnè, le fichier de log prendra par dèfaut le nom de la machine, suffixè avec `-slow.log`. Si un nom de fichier est donnè, mais qu'il manque le chemin, le fichier sera ècrit dans le dossier de donnèes.

Le log de requêtes lentes peut être utilisè pour repèrer les requêtes qui prennent longtemps á s'exècuter, et donc, qui sont candidates á l'optimisation. Avec un grand fichier de log, cela peut devenir difficile. Vous pouvez alors passer le fichier de log á `mysqldumpslow` pour obtenir un sommaire des requêtes dans ce fichier.

Si vous utilisez l'option `--log-long-format` alors les requêtes qui n'utilisent pas d'index sont aussi enregistrèes. Voir Section 4.1.1 [Command-line options], page 192.

4.9.6 Entretien des fichiers de log

Le serveur MySQL peut crèer un grand nombre de fichiers de logs diffèrents, qui permettent de suivre ce qui se passe. Voir Section 4.9 [Log Files], page 327. Vous devez toutefois nettoyer règuliérement ces fichiers, pour être sûr que les logs ne prennent pas tout le disque de la machine.

Lorsque vous utilisez MySQL avec des fichiers de log, vous voudrez, de temps en temps, supprimer ou sauvegarder les fichiers, et demander á MySQL d'utiliser de nouveaux fichiers. Voir Section 4.4.1 [Backup], page 242.

Sous une installation Linux (`Redhat`), vous pouvez utiliser le script `mysql-log-rotate` pour cela. Si vous avez installè MySQL depuis une distribution RPM, le script doit avoir ètè installè automatiquement. Notez que vous devez être prudent avec cette commande si vous utilisez les logs pour la rèplication.

Sur d'autres systémes, vous devez installer un court script par vous même, qui sera exècutè via le dèmon `cron`.

Vous pouvez forcer MySQL á utiliser de nouveaux fichiers de log en utilisant la commande `mysqladmin flush-logs` ou avec la commande SQL `FLUSH LOGS`. Si vous utilisez MySQL version 3.21, vous devez utiliser `mysqladmin refresh`.

Les commandes ci-dessus effectue les tâche suivantes :

- Si le log standard (`--log`) ou le log de requêtes lentes (`--log-slow-queries`) est utilisè, la commande ferme et rèouvre le fichier de log (‘`mysql.log`’ et ‘‘`hostname`‘`-slow.log`’ par dèfaut).
- Si le log de modifications est utilisè (`--log-update`), la commande ferme le log de modification et ouvre un nouveau fichier, avec un nouveau numèro de sèquence plus grand.

Si vous utilisez uniquement le log de modification, vous pour simplement vider les logs sur le disque, et sauver l'ancien fichier de modification dans une sauvegarde. Si vous utilisez le log normal, vous pouvez faire ceci :

```
shell> cd mysql-data-directory
shell> mv mysql.log mysql.old
shell> mysqladmin flush-logs
```

puis faire une sauvegarde du fichier, et le supprimer (‘`mysql.old`’).

4.10 Rèplication de MySQL

Cette section dècrit les diffèrentes fonctionnalitès de la rèplication MySQL. Elle sert de rèfèrence pour les options disponibles avec la rèplication. Vous y trouverez une introduction á la rèplication. Vers la fin, vous y trouverez les questions et problémes les plus frèquents, avec leur solution.

Nous vous suggèrons de visiter notre site web `http://www.mysql.com/` souvent pour y lire les mises á jour de cette section. La rèplication est constamment amèliorèe, et nous modifions souvent le manuel.

4.10.1 Introduction á la rèplication

La rèplication monodirectionnelle est á amèliorer la soliditè et la vitesse de votre application. Pour la soliditè, vous pouvez avoir une copie sur un serveur de sauvegarde, qui prend le relais immèdiatement si le maître rencontre des problémes. L'amèlioration de la rapiditè est obtenue en envoyant les requêtes sans modification sur un serveur qui ne fait que de la lecture. Bien sur, cela ne fonctionnera que si les lectures dominent, mais c'est le cas le plus rèpandu.

Depuis la version 3.23.15, MySQL supporte la rèplication mono directionnelle, en interne. Un serveur sert de maître, et les autres serveurs d'esclaves. Notez qu'un serveur peut servir de maître pour certains, et d'esclaves pour d'autres serveurs. Le maître va conserver un log binaire des modifications (voir Section 4.9.4 [Binary log], page 329) et un fichier d'index des modifications á prendre en compte. L'esclave, au moment de la connexion, informe le

maître oú il s'est arrêtè, rattrape les modifications qui ont eu lieux depuis, puis se bloque en attente des prochaines modifications.

Notez que si vous faites de la rèplication de base, toutes les modifications de cette base devraient être faites sur le maître !

Un autre avantage de la rèplication est que vous pouvez entretenir des sauvegardes immèdiates de vos donnèes, depuis le maître vers l'esclave. Voir Section 4.4.1 [Backup], page 242.

4.10.2 Prèsentation de l'implèmentation de la rèplication

La rèplication MySQL est basèe sur le fait que le serveur va garder la trace de toutes les èvolutions de vos bases (modifications, effacements, etc.) dans un fichier de log binaire (voir Section 4.9.4 [Binary log], page 329) et les esclaves vont venir lire les requêtes du maître dans ce fichier de log, pour pouvoir exècuter les mêmes requêtes sur leurs copies.

Il est **trés important** de comprendre que le fichier de log binaire est simplement un enregistrement des modifications depuis un point fixe dans le temps (le moment oú vous activez le log binaire). Tous les esclaves que vous activez auront besoin de la copie des donnèes qui existaient au moment du dèmarrage du log. Si vous dèmarrez vos esclaves sur sans qu'ils ne disposent des donnèes identiques á celles du maître **au moment du dèmarrage du log binaire**, votre rèplication va èchouer.

Voyez dans la table suivante les indications de compatibilitè entre les maîtres et les esclaves, suivant les versions. Depuis la version 4.0, nous recommandons d'utiliser la même version pour les maîtres et esclaves.

		Maître 3.23.33 et +	**Maître 4.0.0**	**Maître 4.0.1**	**Maître 4.0.3 et +**
Esclave	**3.23.33 et +**	oui	non	non	non
Esclave	**4.0.0**	non	oui	non	non
Esclave	**4.0.1**	oui	non	oui	non
Esclave	**4.0.3 et +**	oui	non	non	oui

Note : MySQL version 4.0.2 n'est pas recommandè pour la rèplication.

Depuis la version 4.0.0, vous pouvez utiliser la commande `LOAD DATA FROM MASTER` pour configurer un esclave. Soyez bien conscient qu'actuellement, `LOAD DATA FROM MASTER` ne fonctionne que si toutes les tables du maître sont du type `MyISAM`, et qu'il est possible d'obtenir un verrou de lecture global, pour qu'aucune lecture ne se fasse durant le transfert des tables depuis le maître. Cette limitation est de nature temporaire, et elle est dûe au fait que nous n'avons pas encore programmè un systéme de sauvegarde des tables sans verrou. La limitation sera supprimèe dans la future version 4.0 une fois que nous aurons programmè le systéme de sauvegarde, qui permettra á `LOAD DATA FROM MASTER` de fonctionner sans bloquer le maître.

Etant donnè la limitation ci-dessus, nous vous recommandons actuellement d'utiliser la commande `LOAD DATA FROM MASTER` uniquement si le jeu de donnèes du maître est petit, ou si un verrou prolongè sur le maître est acceptable. Suivant la vitesse de lecture de `LOAD DATA FROM MASTER` en fonction des systémes, une régle de base indique que le transfert se

fera au rythme de 1 Mo par seconde. Vous pourrez ainsi obtenir une estimation du temps qu'il vous faudra pour transfèrer les donnèes, si le maître et l'esclave sont connectès sur un rèseau de 100 MBit/s, avec des configurations á base de Pentium 700 MHz. Bien sur, votre cas particulier pourra varier en fonction de votre systéme : la régle ci-dessus vous donnera une premiére èvaluation du temps á attendre.

Une fois que l'esclave est correctement configurè, et qu'il fonctionne, il va simplement se connecter au maître et attendre des requêtes de modifications. Si le maître est indisponible ou que l'esclave perd la connexion avec le maître, il va essayer de se reconnecter toutes les `master-connect-retry` secondes jusqu'á ce qu'il soit capable d'ètablir la communication, et de recommencer á appliquer les modifications.

Chaque esclave garde la trace du point oú il en ètait rendu. Le serveur maître n'a pas de notions du nombre d'esclave qui se connectent, ou qui sont á jour á un moment donnè.

La prochaine section explique le processus de rèplication en dètails.

4.10.3 Comment mettre en place la rèplication

Voici les instructions pour mettre en place la rèplication sur votre serveur MySQL. Nous supposons que vous voulez rèpliquer toutes vos bases, et que vous ne l'avez jamais configurè auparavant. Vous aurez besoin d'èteindre briévement le serveur principal pour suivre toutes les instructions.

Si cette mèthode n'est pas la plus simple pour configurer un esclave, ce n'est pas la seule. Par exemple, si vous avez dèjá une sauvegarde des donnèes du maître, et que le maître a dèjá un identifiant de serveur, et le log binaire activè, vous pouvez configurer l'esclave sans èteindre le serveur et sans bloquer les mises á jours. Pour plus de dètails, voyez Section 4.10.7 [Replication FAQ], page 349.

Si vous voulez devenir un vèritable gourou de la rèplication MySQL, nous vous suggèrons de commencer par ètudier, tester et expèrimenter toutes les commandes mentionnèes dans le chapitre Section 4.10.6 [Replication SQL], page 344. Vous devriez aussi vous familiariser avec les options de dèmarrage de '`my.cnf`', Section 4.10.5 [Replication Options], page 338.

1. Assurez vous que vous avez une version rècente de MySQL installèe comme maître et comme esclave.

   Utilisez les versions 3.23.29 et plus rècente. Les anciennes versions utilisaient un autre format de log binaire, et avaient des bogues qui ont ètè corrigès dans les nouvelles versions. Ne nous rapportez pas de bogues avant d'avoir vèrifiè si le probléme est encore prèsent dans la derniére version de MySQL.

2. Crèez un utilisateur MySQL spècial pour la rèplication sur le maître, avec les droits de `FILE` (dans les versions plus anciennes que la versions 4.0.2) ou le droit de `REPLICATION SLAVE` pour les nouvelles versions. Vous devez aussi lui donner les droits de connexion depuis tous les esclaves. Si l'utilisateur ne fait que de la rèplication (ce qui est recommandè), vous n'avez pas á lui donner d'autres droits.

   Par exemple, pour crèer un utilisateur appelè `repl` qui peut accèder au maître, vous pourriez utiliser une commande comme :

   ```
   mysql> GRANT FILE ON *.* TO repl@"%" IDENTIFIED BY '<password>';
   ```

3. Eteignez le serveur MySQL maître.

```
mysqladmin -u root -p<password> shutdown
```

4. Sauvegardez toutes les données de votre serveur maître.

   Le plus simple pour cela (sous Unix) et d'utiliser la commande **tar** pour produire une archive de votre dossier de données total. Le dossier de données dèpend de votre installation.

   ```
   tar -cvf /tmp/mysql-snapshot.tar /path/to/data-dir
   ```

   Les utilisateurs Windows peuvent utiliser le programme `WinZIP` ou un logiciel similaire pour crèer une archive du dossier de données.

5. Dans le fichier '`my.cnf`' du maître, ajouter les options `log-bin` et `server-id=unique number` á la section `[mysqld]` et redèmarrez le serveur. Il est trés important que l'identifiant des esclaves soient diffèrents de celui du maître. Pensez á `server-id` comme á une valeur comparable á une adresse IP : elle identifie de maniére unique un serveur dans la communautè des rèplicateurs.

   ```
   [mysqld]
   log-bin
   server-id=1
   ```

6. Redèmarrez le serveur maître MySQL.

7. Ajoutez les lignes de configuration suivantes dans le fichier '`my.cnf`' des esclaves :

   ```
   master-host=<nom d'hôte du maître>
   master-user=<nom d'utilisateur de rèplication>
   master-password=<mot de passe de l'utilisateur de rèplication>
   master-port=<port TCP/IP du maître>
   server-id=<un identifiant unique, entre 2 et 2^32-1>
   ```

   Remplacez les valeurs entre `<>` avec ce qui correspond á votre systéme.

   `server-id` doit être diffèrent pour chaque serveur qui participe á la rèplication. Si vous spècifiez pas d'identifiant server-id, il sera mis á 1 si vous n'avez pas dèfini de `master-host`, et sinon, il vaudra 2. Notez que dans le cas oú vous omettez le `server-id`, le maître va refuser les connexion des esclaves, et les esclaves vont refuser de se connecter au maître. Par consèquent, omettre `server-id` n'est bon que pour faire des sauvegarde du log binaire.

8. Copiez la sauvegarde des données dans vos esclaves. Assurez vous que les droits sur ces données sont corrects. L'utilisateur qui fait fonctionner MySQL doit avoir les droits d'ècriture et de lecture sur ces fichiers, tout comme le maître l'avait.

9. Redèmarrez les esclaves

Aprés avoir suivi les instructions ci-dessus, les esclaves doivent se connecter au maître, et rattraper les modifications qui ont eu lieu depuis la sauvegarde des données.

Si vous avez oubliè de spècifier un `server-id` pour un esclave, vous allez obtenir l'erreur suivante dans le fichier d'erreur :

```
Warning: one should set server_id to a non-0 value if master_host is set.
The server will not act as a slave.
```

Si vous avez oubliè de le faire pour le maître, les esclaves ne pourront pas se connecter avec le maître.

Si un esclave n'est pas capable de faire la rèplication pour une raison quelconque, vous allez trouvez le message d'erreur dans le fichier de log d'erreurs de l'esclave.

Une fois qu'un esclave fonctionne, vous allez trouver un fichier appelè '`master.info`' dans le même dossier que vos fichiers de logs. Le fichier '`master.info`' est utilisè par l'esclave pour savoir oú il en est rendu dans le log binaire du maître. **Ne le supprimez pas** et ne l'èditez pas, á moins que vous ne sachiez vraiment ce que vous faites. Même dans ce cas, il est prèfèrable d'utiliser la commande `CHANGE MASTER TO`.

4.10.4 Fonctionnalitès de la rèplication et problémes connus

Voici une liste des fonctionnalitès supportèes par la rèplication de MySQL :

- La rèplication s'effectue correctement sur les valeurs `AUTO_INCREMENT`, `LAST_INSERT_ID()` et `TIMESTAMP`.
- `RAND()` dans les commandes `UPDATE` ne se rèplique par correctement. Utilisez `RAND(some_non_rand_expr)` si vous rèpliquez des `UPDATE` avec `RAND()`. Vous pourriez, par exemple, utiliser `UNIX_TIMESTAMP()` comme argument de `RAND()`.
- Vous devez utiliser le même jeu de caractéres (`--default-character-set`) sur le maître et l'esclave. Si ce n'est pas le cas, vous obtiendrez des erreurs de clès doublons dans l'esclave, car une clè qui est indiquèe comme unique sur le maître peut se rèvèler un doublon pour l'esclave.
- En version 3.23, `LOAD DATA INFILE` sera gèrè proprement, tant que le fichier rèside toujours sur le maître au moment de la propagation de la rèplication. Sinon, `LOAD LOCAL DATA INFILE` sera ignorè. En version 4.0, cette limitation a ètè levèe (toutes les formes de `LOAD DATA INFILE` sont correctement rèpliquèes).
- Les requêtes d'`UPDATE` qui utilisent des variables utilisateurs ne sont pas correctement rèpliquèes (pour le moment).
- Les commandes `FLUSH` ne sont pas enregistrèes dans le fichier de log car elle ne sont pas rèpliquèes aux esclaves. Ce n'est pas normalement un probléme, car `FLUSH` ne modifie pas les tables. Cela peut signifier que vous avez modifiè des droits dans les tables MySQL directement sans la commande `GRANT` et que vous avez rèpliquè les droits de `mysql` sans pouvoir faire de commande `FLUSH PRIVILEGES` sur vos esclaves pour les prendre en compte.
- Les tables temporaires sont rèpliquèes depuis la version 3.23.29, á l'exception des cas oú vous èteignez le serveur esclave (et pas juste le thread esclave), que vous avez des tables temporaires ouvertes et qu'elles sont utilisèes dans des modifications ultèrieures. Pour règler ce probléme, utilisez la commande `SLAVE STOP`, vèrifiez la variable de statut `Slave_open_temp_tables` pour vèrifier si elle vaut bien 0, puis exècutez `mysqladmin shutdown`. Si le nombre n'est pas 0, redèmarrez l'esclave avec la commande `SLAVE START` et voyez si vous avez plus de chance la prochaine fois. Il y aura une solution plus èlègante, mais il doit attendre la version 4.0. Dans les anciennes versions, les tables temporaires n'ètaient pas rèpliquèe : nous vous suggèrons de mettre á jour votre version ou d'exècuter la commande `SET SQL_LOG_BIN=0` sur vos clients avant toute requête qui utiliserait les tables temporaires.
- MySQL ne supporte qu'un seul maître, et plusieurs esclaves. En version 4.x, nous allons ajouter un algorithme de sèlection pour que le maître soit modifiè si le maître

initial flanche. Nous allons aussi introduire une notion d'agent dans les processus, pour aider la rèpartition de charge en rèpartissant les requêtes sur les esclaves.

- Depuis la version 3.23.26, il est possible de connecter les serveurs MySQL en chaîne bouclèe (chaque serveur est le maître du prècèdent et l'esclave du suivant, en boucle), avec l'activation de l'option `log-slave-updates`. Notez que de nombreuses requêtes ne vont pas fonctionner dans ce type de configuration á moins que votre code client ne soit ècrit avec beaucoup de soin, pour qu'il se charge des problémes qui pourraient arriver dans diffèrentes sèquences de modifications sur diffèrents serveurs.

  Cela signifie que vous pouvez rèaliser une configuration comme ceci :

  ```
  A -> B -> C -> A
  ```

  Cette configuration va fonctionner si vous ne faites que des modifications non exclusives sur les tables. En d'entre terme, si vous insèrez des donnèes dans les serveurs A et C, vous ne devez pas insèrer une valeur dans le serveur A qui pourrait être en conflit avec une ligne insèrèe dans le serveur C. Vous ne devez pas non plus modifier de ligne sur deux serveurs en même temps, pour èviter les conflits de valeurs.

  Notez que le format de log a ètè modifiè en version 3.23.26 de façon á ce que les esclaves pre-3.23.26 ne soient pas capable de les lire.
- Si la requête sur l'esclave gènére une erreur, le thread esclave s'arrête, et un message sera ajoutè dans le fichier d'erreurs '`.err`'. Vous devriez vous connecter pour corriger manuellement les donnèes de l'esclave, puis relancer l'esclave avec la commande `SLAVE START` (disponible depuis la version 3.23.16. En version 3.23.15, vous devrez redèmarrer le serveur.
- Si la connexion au maître est perdue, l'esclave tente de se reconnecter immèdiatement, et en cas d'èchec, il va retenter toutes les `master-connect-retry` (par dèfaut, 60) secondes. A cause de cela, il est sage d'èteindre le serveur maître et de le redèmarrer règulièrement. L'esclave sera capable de gèrer les problémes rèseau.
- Eteindre l'esclave proprement est sûr, car il garde la trace du point oú il en est rendu. Les extinctions sauvages vont produire des problémes, surtout si le cache disque n'a pas ètè ècrit sur le disque avant que le systéme ne s'arrête. Votre niveau de tolèrance aux pannes sera grandement amèliorè si vous avez de bons onduleurs.
- Si le maître ècoute sur un port non-standard, vous devez aussi le spècifier dans le paramétre `master-port` du fichier '`my.cnf`'.
- En version 3.23.15, toutes les tables et bases seront rèpliquèes. Depuis la version 3.23.16, vous pouvez limiter la rèplication á certaines bases, grâce á l'option `replicate-do-db` du fichier '`my.cnf`' ou simplement d'exclure des bases avec `replicate-ignore-db`. Notez que jusqu'á la version 3.23.23, il y avait un bogue qui ne traitait correctement les commandes `LOAD DATA INFILE` si vous les utilisiez dans une base qui ètaient exclue de la rèplication.
- Depuis la version 3.23.16, `SET SQL_LOG_BIN = 0` va dèsactiver l'enregistrement des donnèes de rèplication (binaire) sur le maître, et `SET SQL_LOG_BIN = 1` la rèactivera. Vous aurez besoin des droits de `SUPER` (en MySQL 4.0.2 et plus rècent), ou `PROCESS` (dans les anciens droits MySQL) pour faire cela.
- Depuis la version 3.23.19, vous pouvez nettoyer les restes de la rèplication lorqu'un probléme est survenu avec les commandes `FLUSH MASTER` et `FLUSH SLAVE`. En version

3.23.26, nous avons renommè cette commande en `RESET MASTER` et `RESET SLAVE` respectivement, pour clarifier leur action. Les anciennes variantes de `FLUSH` fonctionnent encore, pour la compatibilitè.

- Depuis la version 3.23.23, vous pouvez modifier les maîtres et ajuster la position de log avec la commande `CHANGE MASTER TO`.
- Depuis la version 3.23.23, vous pouvez dire au maître que les modifications de certaines bases ne doivent pas être enregistrèes dans le fichier de log binaire, avec l'option `binlog-ignore-db`.
- Depuis la version 3.23.26, vous pouvez utiliser l'option `replicate-rewrite-db` pour dire á l'esclave d'appliquer les modifications d'une base dans une autre.
- Depuis la version 3.23.28, vous pouvez utiliser la commande `PURGE MASTER LOGS TO 'log-name'` pour vous dèbarasser des anciens logs sur l'esclave. Cela va supprimer les vieux logs, maisp as le log appelè `'log-name'`.
- Etant donnè la nature non transactionnelle des tables MySQL, il est possible qui va ne faire qu'une partie de la modification, et retourner une erreur. Cela peut arriver, par exemple, dans une insertion multiple dont une des lignes viole une contrainte d'unicitè, ou si un trés long `UPDATE` est interrompu au milieu du stock de ligne. Si cela arrive sur le maître, l'esclave va s'arrêter et attendre que l'administrateur dècide quoi faire, á moins que l'erreur soit lègitime, et que la requête arrive á la même conclusion. Si le code d'erreur n'est pas dèsirable, certaines erreurs (voire même toutes), peuvent être masquèes avec l'option `slave-skip-errors`, depuis la version 3.23.47.
- Alors que des tables particuliéres peuvent être exclues de la rèplication avec les options `replicate-do-table`/`replicate-ignore-table` ou `replicate-wild-do-table`/`replicate-wild-ignore-table`, il y a actuellement des problémes de conception, qui provoque des rèsultats inattendus dans certains cas trés rares. Le protocole de rèplication n'informe pas explicitement l'esclave quelles tables vont être modifièes par la requête : l'esclave doit donc analyser lui-même la requête pour le savoir. Pour èviter une analyse rèpètitive des requêtes qui seront finalement exècutèes, l'exclusion de table est actuellement implèmentèes en envoyant la requête á l'analyseur MySQL, qui va court circuiter la requête, et qui indiquera la rèussite si une table utilisèe doit être ignorèe. En plus de plusieurs sources de ralentissement, cette approche va fatalement engendrer des bogues, et il en existe dux depuis la version 3.23.49 – car l'analyseur ouvre automatiquement la table lors de l'analyse de la requête, la table qui doit être ignorèe doit donc exister sur l'esclave. L'autre bogue est que si le thread esclave modifie partiellement la table á ignorer, le thread ne va pas rèaliser que la table aurait du être ignorèe, et va suspendre la rèplication. Alors que le premier bogue est conceptuellement simple á corriger, nous n'avons pas encore trouvè de moyen pour corriger le second sans intervenir massivement dans le code, et donc, en compromettre la stabilitè. Il existe un palliatif pour les deux, dans les cas rares ou cela arrive á votre application : utilisez `slave-skip-errors`.

4.10.5 Options de rèplication dans le fichier '`my.cnf`'

Si vos utilisez la rèplication, nous vous recommandons d'utiliser MySQL version 3.23.30 ou plus rècent. Les vieilles versions fonctionneront, mais elles prèsentent des bogues et

manquent les nouvelles fonctionnalitès. Certaines des options qui seront prèsentèes ici ne seront pas forcèment disponibles dans votre version de MySQL, si vous n'utilisez pas la version la plus rècente. Pour toutes les options spècifiques á la version 4.0, il y a une note qui l'indique. Sinon, si vous dècouvrez une option qui vous interesse mais qui n'est pas disponible dans votre version 3.23, et que vous en avez vraiment besoin, mettez á jour votre installation.

Notez bien que la version 4.0 est toujours en phase alpha, et que certaines fonctionnalitès ne fonctionneront peut être pas aussi bien que prèvu. Si vous voulez vraiment essayer de nouvelles fonctionnalitès de la version 4.0, nous vous recommandons de le faire de façon á ce que cela ne perturbe pas vos applications critiques.

Sur le maître et l'esclave, vous devez utiliser l'option `server-id`. Elle doit reprèsenter un identifiant unique de rèplication. Vous devriez choisir une valeur unique entre 1 et 2^32-1 pour chaque maître et esclave. Example: `server-id=3`

La table suivante dècrit les options que vous pouvez utiliser avec `MASTER` :

Option	Description
`log-bin=filename`	Ecrit le fichier de log binaire á la position indiquèe. Notez que vis vous donner un paramétre avec une extension (par exemple, `log-bin=/mysql/logs/replication.log`), les versions jusqu'en 3.23.24 ne fonctionneront pas correctement avec la rèplication, si vous utilisez la commande `FLUSH LOGS`. Le probléme a ètè corrigè avec la version 3.23.25. Si vous utilisez ce type de nom de fichier de log, `FLUSH LOGS` sera ignorè sur les logs binaires. Pour nettoyer le log, exècutez la commande `FLUSH MASTER`, et n'oubliez pas la commande `FLUSH SLAVE` sur chacun des esclaves. En version 3.23.26 et plus rècent, vous devriez utiliser `RESET MASTER` et `RESET SLAVE`.
`log-bin-index=filename`	Comme l'utilisateur peut exècuter une commande `FLUSH LOGS`, nous devons savoir quel fichier de log est actuellement actif, lequels ont ètè dèjá utilisès, et dans quel ordre. Cette information est stockèe dans le fichier d'index de log. Par dèfaut, ce fichier s'appelle '`hostname`.index'. Nous n'avez pas á le modifier. Example: `log-bin-index=db.index`
`sql-bin-update-same`	Si cette option est activèe, en donnant une valeur á `SQL_LOG_BIN` vous donnerez automatiquement la même valeur á `SQL_LOG_UPDATE` et vice-versa.

`binlog-do-db=database_name`	Indique au maître qu'il doit mettre les modifications dans le fichier de log binaire, si la base courante est la base `database_name`. Toutes les autres bases sont ignorèes. Notez que vous si vous utilisez cette option, vous devez vous assurer que vous ne faites des modifications que dans la base courante. Example: `binlog-do-db=sales`
`binlog-ignore-db=database_name`	Indique au maître que les modifications dont la base courante est `database_name` ne doivent pas être stockèes dans le log binaire. Notez que vous si vous utilisez cette option, vous devez vous assurer que vous ne faites des modifications que dans la base courante. Example: `binlog-ignore-db=accounting`

La table suivante dècrit les options que vous pouvez utiliser avec `SLAVE` :

Option	**Description**
`master-host=host`	Le nom d'hôte du maître, ou l'adresse IP de rèplication. Si cette valeur n'est pas spècifièe, l'esclave ne dèmarrera pas. Notez que la configuration de `master-host` sera ignorèe si il existe un fichier '`master.info`' valide. Il est probable qu'un nom mieux choisi pour cette option aurait ètè `bootstrap-master-host`, mais il est trop tard aujourd'hui. Example: `master-host=db-master.mycompany.com`
`master-user=username`	Le nom d'utilisateur que l'esclave va utiliser pour s'identifier lors de la connexion au maître. Cet utilisateur doit avoir les droits de `FILE`. S'utilisateur maître n'est pas configurè, l'utilisateur `test` est utilisè. La valeur dans '`master.info`' aura prioritè, si elle peut être lue. Example: `master-user=scott`
`master-password=password`	Le mot de passe que l'esclave va utiliser automatiquement avec le maître. Si il n'est pas configurè, le mot de passe vide sera utilisè. La valeur dans '`master.info`' aura prioritè, si elle peut être lue. Example: `master-password=tiger`
`master-port=portnumber`	Le port sur lequel le maître ècoute. Si cette valeur n'est pas configurèe, la valeur compilèe de `MYSQL_PORT` est utilisèe. Si vous n'avez pas jouè avec les options du script `configure`, ce devrait être le port 3306. La valeur dans '`master.info`' aura prioritè, si elle peut être lue. Example: `master-port=3306`

`master-connect-retry=seconds` Le nombre de secondes que le thread esclave va attendre avant de tenter á nouveau de se connecter au maître, dans le cas oú le maître n'est plus joignable. Par dèfaut, c'est 60.
Example: `master-connect-retry=60`

`master-ssl` Disponible aprés la version 4.0.0. Active le mode SSL pour la rèplication. Soyez prèvenu que c'est une fonctionnalitè trés rècente.
Example: `master-ssl`

`master-ssl-key` Disponible aprés la version 4.0.0. Le nom du fichier de clè maître SSL. Uniquement valable si l'option `master-ssl` est active.
Example: `master-ssl-key=SSL/master-key.pem`

`master-ssl-cert` Disponible aprés la version 4.0.0. Le nom du fichier de certificat SSL. Uniquement valable si l'option `master-ssl` est active.
Example: `master-ssl-key=SSL/master-cert.pem`

`master-info-file=filename` La position du fichier qui enregistre la derniére opèration faite avec le maître, durant la rèplication. Le fichier par dèfaut est '`master.info`' dans le dossier de donnèes. Vous n'avez pas besoin de modifier cela.
Example: `master-info-file=master.info`

`report-host` Disponible aprés la version 4.0.0. Le nom d'hôte ou l'adresse IP de l'esclave, qui doit être enregistrèe lors de l'identification. Cette valeur apparaitra dans le rèsultat de la commande `SHOW SLAVE HOSTS`. Laissez la vide si vous ne souhaitez pas que l'esclave s'enregistre lui-même. Notez qu'il n'est pas suffisant pour le maître de simplement lire l'IP de l'esclave dans la socket de connexion. A cause du `NAT` et d'autres problémes de routage, cette IP peut ne pas êtra valable pour la connexion du maître sur l'esclave.
Example: `report-host=slave1.mycompany.com`

`report-port` Disponible aprés la version 4.0.0. Port de connexion á l'esclave indiquè au maître durant l'enregistrement. Utilisez cette option uniquement si l'esclave doit ècouter sur un port non standard, ou si vous avez un tunnel particulier pour le maître. En cas de doute, laissez le vide.

`replicate-do-table=db_name.table_name`

Restreint la rèplication á certaines tables. Pour spècifier plus d'une table, vous pouvez utiliser cette directive plusieurs fois, une pour chaque table. Cette option va fonctionner pour les modifications inter-bases, contrairement á `replicate-do-db`.
Example: `replicate-do-table=some_db.some_table`

`replicate-ignore-table=db_name.table_name`

Indique á l'esclave d'ignorer les commandes qui modifie les tables spècifièes (même si d'autres tables de la même commande sont aussi modifièes). Pour spècifier plus d'une table, vous pouvez utiliser cette directive plusieurs fois, une pour chaque table. Cette option va fonctionner pour les modifications inter-bases, contrairement á `replicate-ignore-db`.
Example: `replicate-ignore-table=db_name.some_table`

`replicate-wild-do-table=db_name.table_name`

Restreint la rèplication á certaines tables qui vèrifient le masque d'expression règuliére indiquè. Pour spècifier plus d'une table, vous pouvez utiliser cette directive plusieurs fois, une pour chaque table. Cette option va fonctionner pour les modifications inter-bases.
Example: `replicate-wild-do-table=foo%.bar%` va rèpliquer uniquement les modifications qui utilisent une table dont le nom de base commence par `foo`, et le nom de table commence par `bar`.

`replicate-wild-ignore-table=db_name.table_name`

Interdit la rèplication á certaines tables qui vèrifient le masque d'expression règuliére indiquè. Pour spècifier plus d'une table, vous pouvez utiliser cette directive plusieurs fois, une pour chaque table. Cette option va fonctionner pour les modifications inter-bases.
Example: `replicate-wild-ignore-table=foo%.bar%` va ignorer la rèplication des modifications qui utilisent une table dont le nom de base commence par `foo`, et le nom de table commence par `bar`.

`replicate-ignore-db=database_name`

Restreint la rèplication aux bases de données indiquèes. Pour spècifier plus d'une base, vous pouvez utiliser cette directive plusieurs fois, une pour chaque base. Vous ne devez pas utiliser cette option si vous faites des modifications inter-bases et que vous ne voulez pas que ces modifications soient prises en compte.

La principale raison de ce comportement est qu'il est difficile d'utiliser uniquemnt la ligne de commande pour savoir si une requête doit être rèpliquèe ou pas. Par exemple, si vous utilisez une commande d'effacement multi-table en MySQL 4.x, qui utilise plusieurs bases. Il est aussi trés facile de vèrifier la base courante, car cela ne s'applique qu'une fois au moment de la connexion, ou d'un changement de base.

Si vous voulez que des modifications inter-bases fonctionnent, assurez vous que vous avez la version 3.23.28 ou plus rècente, et utilisez `replicate-wild-ignore-table=db_name.%`.

Example: `replicate-ignore-db=some_db`

`replicate-do-db=database_name`

Restreint la rèplication aux commandes dont le nom de base courante est `database_name`. Pour spècifier plus d'une base, vous pouvez utiliser cette directive plusieurs fois, une pour chaque base. Notez que cette option ne va pas autoriser la rèplication de requêtes inter-bases telles que `UPDATE some_db.some_table SET foo='bar'` lorsque vous avez sèlectionnè une autre base (ou qu'aucune base n'est sèlectionnèe). Si vous avez besoin de faire des modifications inter-bases, assurez vous que vous avez la version 3.23.28 ou plus rècente, et utilisez `replicate-wild-do-table=db_name.%`.

Example: `replicate-do-db=some_db`

`log-slave-updates`

Indique á l'esclave d'enregistrer les modifications du thread esclave dans le log binaire. Par dèfaut, cette option est inactive (off). Vous aurez besoin de cette fonctionnalitè si vous installez les esclaves en cascade.

`replicate-rewrite-db=from_name->to_name`

Reporte les modifications qui ont lieu dans une base, dans une autre.

Example: `replicate-rewrite-db=master_db_name->slave_db_name`

`slave-skip-errors= [err_code1,err_code2,... | all]`

Disponible depuis la version 3.23.47 et plus rècent. Indique au thread esclave de continuer la rèplication si une requête gènére une erreur issue de la liste fournie. Normalement, la rèplication va s'arrêter dés qu'une erreur survient, pour que l'utilisateur puisse corriger les incohèrence manuellement. N'utilisez pas cette option si nous ne comprenez pas complétement les raisons qui gènérent ces erreurs. Si il y n'a pas de bogues dans vos configurations de rèplication, et pas de bogues dans MySQL, vous ne devriez jamais interrompre la rèplication. Utiliser cette option á tort et á travers va dèsynchroniser les esclaves et vous n'aurez aucune idèe des problémes qui sont survenus.

Pour les codes d'erreurs, vous devez utiliser les numèros d'erreurs qui sont fournis dans le log d'erreur de l'esclave, et dans le rèsultat de la commande `SHOW SLAVE STATUS`. La liste compléte des message d'erreurs sont disponibles dans le fichier source '`Docs/mysqld_error.txt`'.

Vous pouvez (mais ne devriez pas) aussi utiliser la valeur hautement dangeureuse de `all` (tous), pour ignorer toutes les erreurs de rèplication, et continuer comme si de rien n'ètait. Inutile de vous dire que si vous l'utilisez, nous ne pouvons rien garantir quand á l'intègritè de vos donnèes. Ne venez pas vous plaindre que les donnèes de vos esclaves sont trés diffèrentes des donnèes du maître : vous aurez ètè prèvenus.

Exemple :
`slave-skip-errors=1062,1053` or `slave-skip-errors=all`

`skip-slave-start`

Indique au serveur esclave de ne pas dèmarrer l'esclave au dèmarrage. L'utilisateur le dèmarrera manuellement plus tard, avec `SLAVE START`.

`slave_compressed_protocol=#`

Si cette option vaut 1, alors le protocole compressè est utilisè entre le maître et l'esclave, si les deux le supporte.

`slave_net_timeout=#`

Le nombre de secondes d'attente de donnèes depuis le maître, avant d'abandonner la lecture.

4.10.6 Commandes SQL liées á la rèplication

La rèplication peut être contrôlèe depuis l'interface SQL. Voici un rèsumè des commandes disponibles :

Commande	Description

`SLAVE START`	Dèmarre le thred esclave. Depuis MySQL 4.0.2, vous pouvez ajouter les options `IO_THREAD` ou `SQL_THREAD` á cette commande, pour dèmarrer le thread I/O ou le thread SQL. Le thread I/O lit les requêtes depuis le maître, et les stockent dans le log de relais. Le thread SQL lit le log de relais, et exècute les requêtes. (Esclave)
`SLAVE STOP`	Stoppe le thread esclave. Comme `SLAVE START`, cette commande peut être utilisèe avec les options `IO_THREAD` et `SQL_THREAD`. (Esclave)
`SET SQL_LOG_BIN=0`	Dèsactive le log binaire, si l'utilisateur a les droits de `SUPER`. Ignorè sinon. (Maître)
`SET SQL_LOG_BIN=1`	Rèactive le log binaire, si Re-enables update logging, si l'utilisateur a les droits de `SUPER`. Ignorè sinon. (Maître)
`SET GLOBAL SQL_SLAVE_SKIP_ COUNTER=n`	Ignore les `n` prochains èvènements du maître. Uniquement valide lorsque le thread esclave ne fonctionne pas, sinon une erreur est affichèe. Trés pratique pour corriger des petits problémes de rèplications.
`RESET MASTER`	Efface tous les logs binaires dans le fichier d'index, et vide le fichier de log binaire. Dans les versions antèrieures á la 3.23.26, utilisez `FLUSH MASTER`. (Maître)
`RESET SLAVE`	Efface la position de rèplication de l'esclave dans les logs du maître. Makes the slave forget its replication position in the master logs. Dans les versions antèrieures á la 3.23.26, cette commande s'appelait `FLUSH SLAVE`. (Esclave)
`LOAD TABLE tblname FROM MASTER`	Tèlècharge une copie des tables depuis le maître vers l'esclave. Implèmentèe principalement pour le dèbogage de `LOAD DATA FROM MASTER`, mais certains utilisateurs "gourmets" pourront la trouver pratique pour d'autres utilisations. Ne l'utilisez pas si vous vous considèrez comme un utilisater "non-hacker". (Esclave)

`LOAD DATA FROM MASTER`

Disponible depuis 4.0.0. Prend une copie des données du maître et les transfére vers l'esclave. Met á jour les valeurs de `MASTER_LOG_FILE` et `MASTER_LOG_POS` de maniére á ce que l'esclave dèmarre la rèplication depuis la position correcte. Cette commande respecte les régles d'exclusion des tables et des bases spècifiées dans les options `replicate-*`. Actuellement fonctionne uniquement avec les tables `MyISAM` et pose un verrou global en lecture sur toutes les tables, pour faire la sauvegarde. Dans le futur, il est prèvu de rendre cette commande compatible avec les tables `InnoDB` et de supprimer le besoin du verrou global pour faire une sauvegarde non bloquante.

`CHANGE MASTER TO master_def_list`

Change les paramétres du maître avec les valeurs spècifièes dans l'argument `master_def_list` et redèmarre le thread esclave. `master_def_list` est une liste sèparèe par des virgules, avec les variables suivantes : `MASTER_HOST`, `MASTER_USER`, `MASTER_PASSWORD`, `MASTER_PORT`, `MASTER_CONNECT_RETRY`, `MASTER_LOG_FILE` et `MASTER_LOG_POS`. Par exemple :

```
CHANGE MASTER TO
  MASTER_HOST='master2.mycompany.com',
  MASTER_USER='replication',
  MASTER_PASSWORD='bigs3cret',
  MASTER_PORT=3306,
  MASTER_LOG_FILE='master2-bin.001',
  MASTER_LOG_POS=4;
```

Vous n'avez pas besoin de spècifier les valeurs que vous ne souhaitez pas changer. Les valeurs omises conserveront leur valeur actuelle, même si vous modifiez l'hôte ou le port. Dans ce cas, l'esclave supposer que puisque vous vous connectez á un nouvel hôte ou un nouveau port, le maître est diffèrent. Par consèquent, il les anciennes valeurs de log et position ne sont plus applicables, et il va automatiquement leur donner la valeur de la chaîne vide, et de 0, respectivement, qui sont les valeurs de dèmarrage. Notez qui si vous redèmarrez l'esclave il va se souvenir de son ancien maître. Si vous ne le souhaitez pas, il suffit d'effacer le fichier '`master.info`' avant de le redèmarrer, et l'esclave va lire la valeur depuis le fichier '`my.cnf`' ou la ligne de commande.

Cette commande est pratique pour configurer un esclave lorsque vous avez une sauvegarde du maître, et que vous avez sauvè la valeur du log et de son offset chez le maître. Vous pouvez alors exècuter la commande `CHANGE MASTER TO MASTER_LOG_FILE='log_name_on_master', MASTER_LOG_POS=log_offset_on_master` sur l'esclave aprés avoir restaurè la sauvegarde. (Esclave)

`SHOW MASTER STATUS`

Fournit les informations de statut sur le point de contrôle du fichier de log binaire du maître. (Maître)

`SHOW SLAVE HOSTS`	Disponible depuis la version 4.0.0. Cette commande liste les esclaves actuellement enregistrè auprés du maître. (Maître)
`SHOW SLAVE STATUS`	Fournit les informations essentielles sur le statut de l'esclave. (Esclave)
`SHOW MASTER LOGS`	Uniquement disponible depuis la version 3.23.28. Liste les logs binaires sur le maître. Vous devriez utiliser cette commande avant d'utiliser la commande `PURGE MASTER LOGS TO` pour savoir lesquels vous voulez effacer. (Maître)
`SHOW BINLOG EVENTS [ IN 'logname' ] [ FROM pos ] [ LIMIT [offset,] rows ]`	Affiche les èvènements dans le log binaire. Initialement utilisè pour le test et le dèbogage dans le log binaire, mais il peut aussi être utilisè par les clients habituels qui, pour une raison ou une autre, ont besoin de lire le contenu du log binaire. (Maître)
`SHOW NEW MASTER FOR SLAVE WITH MASTER_LOG_FILE='logfile' AND MASTER_LOG_POS=pos AND MASTER_LOG_SEQ=log_seq AND MASTER_SERVER_ID=server_id`	Cette commande est utilisèe lorsque l'esclave d'un maître probablement mort ou indisponible doit être redirigè vers un autre maître qui fait la rèplication du même maître. La commande va retourner les nouvelles coordonnèes de rèplication (le nom du point de contrôle dans le fichier de log binaire et son offset). Le rèsultat de la commande peut être utilisè dans un appel á la commande `CHANGE MASTER TO`. Les utilisateurs normaux ne doivent jamais exècuter cette commande. Elle est rèservèe á l'utilisation interne par le code de rèplication sans erreur. Nous risquons de changer cette syntaxe, si vous trouvons un autre moyen plus intuitif de faire cette opèration.

`PURGE MASTER LOGS TO 'logname'` Disponible depuis la version 3.23.28. Efface tous les logs de rèplication qui sont listès dansle fichier d'index de rèplication, comme ètant des logs dèjá utilisès. Le log qui est spècifiè devient alors log courant. Exemple :

```
PURGE MASTER LOGS TO 'mysql-bin.010'
```

Cette commande ne fait rien et èchoue avec une erreur si vous avez un esclave actif qui est actuellement en train de lire un des fichiers de logs que vous essayez d'effacer. Cependant, si vous avez un esclave qui dort, et que vous effacez un des logs qu'il voudra lire plus tard, l'esclave ne pourra pas effectuer la rèplication lorsque son temps arrivera. La commande est sûre lorsque d'autres esclaves effectue la rèplication : vous n'aurez pas á les arrêter.

Vous devez d'abord vèrifier tous les esclaves, avec la commande `SHOW SLAVE STATUS` pour voir quels logs ils utilisent, pour faire une liste des logs sur le maître avec la commande `SHOW MASTER LOGS`, trouver le plus vieux log de tous (si tous les esclaves sont mis á jours, cela doit être le dernier de la liste), effectuer une sauvegarde optionnelle de tous les logs, puis supprimer les logs du serveur.

4.10.7 FAQ de la rèplication

Q : Comment puis-je configurer un esclave si le maître fonctionne dèjá, et que je ne veux pas le stopper?

R : Il y a plusieurs solutions. Si vous avez effectuè une sauvegarde du maître á un moment et enregistrè le nom et l'offset du binlog (issu du rèsultat de la commande `SHOW MASTER STATUS`) correspondant á la sauvegarde, faites ceci :

- Assurez vous qu'un identifiant unique est assignè á l'esclave.
- Exècutez la commande `CHANGE MASTER TO MASTER_HOST='master-host-name', MASTER_USER='master-user-name', MASTER_PASSWORD='master-pass', MASTER_LOG_FILE='recorded-log-name', MASTER_LOG_POS=recorded_log_pos`
- Exècutez la commande `SLAVE START`

Si vous n'avez pas de copie de sauvegarde, voici un moyen rapide d'en faire une :

- `FLUSH TABLES WITH READ LOCK`
- `gtar zcf /tmp/backup.tar.gz /var/lib/mysql` (ou toute variation de cette commande)
- `SHOW MASTER STATUS` - assurez vous de noter le rèsultat. Vous en aurez besoin ultèrieurement.

- `UNLOCK TABLES`

Aprés cela, suivez les instructions comme pour le cas oú vous avez dèjá votre sauvegarde, et que vous avez enregistrè le nom et l'offset du point de contrôle du log binaire. Tant que les logs binaires du serveur sont toujours lá, vous allez pouvoir rattrapper tout ce qui se fait sur le serveur principal. Vous pourriez même attendre plusieurs jours ou mois avant de mettre en place votre esclave. En thèorie, le temps d'attente peut être infini. En pratique, les limitations sont l'espace disque du maître, et le temps que cela prendra á l'esclave pour rattrapper le temps.

En version 4.0.0 et plus rècentes, vous pouvez aussi utiliser `LOAD DATA FROM MASTER`. C'est une commande pratique pour faire une copie de la base, l'envoyer á l'esclave, et ajutser le point de contrôle du log binaire, tout en une seule commande. Dans le future, `LOAD DATA FROM MASTER` sera la mèthode recommandèe pour configurer un esclave. Soyez prèvenus, que le verrou de lecture posè par la commande sur le serveur peut rester en place un trés long moment, si vous utilisez cette commande : elle n'est pas encore implèmentèe de maniére efficace. Si vous avez de grandes tables, prèfèrez donc la mèthode qui utilise la sauvegarde via l'utilitaire `tar` aprés avoir exècutè la commande `FLUSH TABLES WITH READ LOCK`.

Q : Est ce que l'esclave doit être connectè en permanance au serveur?

R : Non, il n'est pas obligè. Vous pouvez èteindre l'esclave et le laisser dèconnecter plusieurs heures ou jours, puis le reconnecter pour le voir rècupèrer les modifications et rattrapper le temps. Puis, se dèconnecter á nouveau. De cette façon, vous pouvez, par exemple, configurer un esclave via une connexion modem, qui n'utilise que de bréve pèriode de connexions. L'implication de cela est qu'il n'est jamais garantit que l'esclave soit synchronisè avec le maître, á moins que vous ne preniez des mesures pour cela. Dans le futur, nous allons avoir l'option de bloquer le maître jusqu'á ce que au moins un des esclaves soit synchronisè.

Q : Comment puis-je forcer le maître á bloquer les modifications jusqu'á ce que l'esclave ait tout rattrappè?

R : Exècutez les commandes suivantes :

- Master: `FLUSH TABLES WITH READ LOCK`
- Master: `SHOW MASTER STATUS` - notez le nom du point de contrôle et son offset.
- Esclave : `SELECT MASTER_POS_WAIT('recorded_log_name', recorded_log_offset)` Lorsque la sèlection est terminèe, l'esclave est synchronisè avec le maître.
- Maître : `UNLOCK TABLES` - Le maître reprend ses oprèations.

Q : Pourquoi est ce que je vois parfois plusieurs threads `Binlog_Dump` sur le maître, une fois que j'ai redèmarrè un esclave?

R : `Binlog_Dump` est un processus continu, qui est gèrè par le serveur de cette maniére :

- Rattrappe les modifications.
- Une fois qu'il n'y a plus de modifications, va dans `pthread_cond_wait()`, pour que nous puissions être prèvenus d'une modificaiton ou d'un arrêt.
- Au moment de l'alerte, vèrifie la raison. Si nous ne sommes pas supposès nous arrêter, contine la boucle `Binlog_dump`.
- Si il y a une erreur fatale, par exemple la dètection d'un client mort, termine la boucle.

Si le thread de l'esclave s'arrête sur l'esclave, le thread correspondant `Binlog_Dump` sur le maître ne va pas le remarquer jusqu'á la prochaine modification sur le serveur (ou mort

de processus), qui est nècessaire pour l'alerter, via `pthread_cond_wait()`. Dans le même temps, l'esclave peut avoir ouvert une autre connexion, qui a gènèrè un autre thread `Binlog_Dump`.

Le probléme ci-dessus ne doit plus être prèsent depuis a version 3.23.26 et plus rècent. En version 3.23.26, nous avons ajoutè un identifiant `server-id` pour chaque serveur de rèplication et dèsormais, les vieux zombies sont tuès sur le maître, lorsqu'une nouvelle connexion de rèplication se connecte sur le maître avec le même identifiant d'esclave.

Q : Comment faire tourner les logs?

R : En version 3.23.28, vous devez utiliser la commande `PURGE MASTER LOGS TO` aprés avoir dèterminè quels logs peuvent être effacès, et optionnellement, avoir fait la sauvegarde des premiers. Dans les versions plus anciennes, le processus ètait bien plus douloureux, et ne pouvait être rèalisè sans arrêter tous les esclaves, pour pouvoir rèutiliser les noms des points de contrôle. Vous devez stopper les threads esclaves, èditer le fichier d'index de log á la main, effacer tous les vieux logs, redèmarrer le serveur, redèmarrer les esclaves, puis effacer les fichiers de logs.

Q : Comment puis-je passer á une configuration de rèplication á chaud?

R : Si vous faîtes une mise á jour depuis les versions antèrieures á la 3.23.26, vous devez simplement verrouiller les tables maîtres, laisser les esclaves rattrapper le temps, puis utiliser la commande `FLUSH MASTER` sur le master, et `FLUSH SLAVE` sur l'esclave pour redèmarrer les logs, et redèmarrer la nouvelle version du maître et de l'esclave. Notez que l'esclave peut être inactif pour un certain temps : comme le maître note toutes les modifications, l'esclave sera capable de rattrapper, dés qu'il peut se reconnecter.

Depuis la version 3.23.26, nous avons verrouillè le protocole de rèplication pour les modifications, ce qui fait que vous pouvez modifier le maître et les esclaves á la volèe vers une nouvelle version de la sèrie des 3.23 et vous pouvez même avoir diffèrentes versions de MySQL comme esclave et comme maître, tant qu'ils sont plus rècents que la 3.23.26.

Q : Quels sont vos conseils concernant la rèplication bi-bidirectionnelle?

R : La rèplication MySQL ne supporte aucun protocole de verrouillage entre le maître et l'esclave pour garantir l'atomicitè d'une modification entre les serveurs. En d'autres termes, il est possible pour un client A de faire une modification sur le serveur 1 et que dans le même temps, avant que cela ne se soit propagè au serveur 2, un client B se connecte au serveur 2, et fasse une modification sur le serveur 2 qui ne dèbouchera pas sur le même ètat que celui dans lequel le serveur 1 est. C'est ainsi qu'il ne faut pas lier de cette façon deux serveurs, á moins que les modifications ne puisse se faire dans n'importe quel ordre, ou que vous sachiez prendre en charge des modifications anarchiques.

Vous devez aussi rèaliser que la rèplication bi-directionnelle n'amèliore pas beaucoup les performances, tout au moins au niveau des modifications. Les deux serveurs doivent faire la même quantitè de modifications, ainsi qu'un serveur seul le ferait. La seule diffèrence est qu'il va y avoir moins de verrous, car les modifications qui proviennent d'un autre serveur seront optimisè par l'esclave. Cet avantage peut aussi être annulè par les dèlais rèseau.

Q : Comment puis-je utiliser la rèplication pour amèliorer les performances de mon systéme ?

R : Vous devez configurer un serveur en maître et y diriger toutes les ècritures, puis configurer les autres en esclaves dans la limite de vos moyens, et y distribuer les lectures. Vous

pouvez aussi dèmarrer les esclaves en mode `--skip-bdb`, `--low-priority-updates` et `--delay-key-write=ALL` pour accèlèrer les esclaves. Dans ce cas, l'esclave va utiliser les tables non transactionnelles `MyISAM` au lieu des tables `BDB` pour obtenir plus de vitesse.

Q : Que dois-je faire pour prèparer mon code client á la rèplication?

R : Si la partie de votre code qui rèalise les accés aux bases de donnèes a ètè proprement modularisèe, la convertir en une configuration qui supporte la rèplication ne sera pas un probléme : modifiez simplement votre base pour qu'elle aille lire sur les esclaves et le maître, mais ne fasse que des modifications avec le maître. Si votre code n'a pas ce niveau d'abstraction, l'installation du systéme de rèplication vous donnera alors la motivation ou la raison pour le faire. Vous devriez commencer par crèer une couche d'abstraction ou un module avec les fonctions suivantes :

- `safe_writer_connect()`
- `safe_reader_connect()`
- `safe_reader_query()`
- `safe_writer_query()`

`safe_` signifie que la fonction devra prendre en charge toutes les conditions d'erreurs.

Vous devriez alors convertir votre code client pour qu'il utilise cette librairie. Cela peut être un processus laborieux et dèroutant, mais il va s'avèrer payant dans le long terme. Toutes les applications qui suivent la technique ci-dessus pourront alors prendre avantage des solutions de rèplication. Le code sera aussi bien plus facilement entretenu, et ajouter des options sera trivial. Vous devrez modifier une ou deux fonctions, comme par exemple pour enregistrer le temps de calcul de certaines requêtes, ou les requêtes qui vous retournent des erreurs. Si vous avez ècrit beaucoup de code jusqu'ici, vous pourriez vouloir automatiser la conversion en utilisant l'utilitaire de Monty, `replace`, qui est distribuè avec la distribution standard de MySQL, ou bien simplement en ècrivant un script Perl. Avec un peu de chance, votre code suit des conventions connues. Si ce n'est pas le cas, alors vous serez peut être conduit á rèècrire votre application de toutes maniéres, ou bien, á lui appliquer des mèthodes á la main.

Notez que, bien sur, vous pouvez utiliser diffèrents nom pour les fonctions. Ce qui est important, c'est que vous ayez des interfaces unifièes pour vous connecter en lecture ou en ècriture, et pour exècuter des lectures et ècritures.

Q : Quand et combien de rèplications de MySQL permettront d'amèliorer les performances de mon systéme?

R : La rèplication MySQL est particuliérement avantageuse pour les systémes qui gérent des lectures frèquentes, et des ècritures plus rares. En thèorie, en utilisant uniquement un maître et beaucoup d'esclaves, vous pouvez augmenter les performances de votre systéme jusqu'á saturation de la bande passante ou du maître, pour les modifications.

Afin de dèterminer le nombre d'esclaves que vous pouvez obtenir voir les performances de votre systéme s'amèliorer, vous devez bien connaître les types de requêtes que vous utilisez, et empiriquement dèterminer la relation entre le nombre de lectures et d'ècritures (par secondes, ou maximum absolu), pour un maître et un esclave. L'exemple ci-dessous va vous montrer comment faire des calculs simples.

Imaginons que votre charge systéme soit constituèe de 10% d'ècriture et de 90% de lectures. Nous avons aussi dèterminè que le maximum de lectures `max_reads` = 1200 - 2 * `max_`

`writes`, ou, en d'autres mots, notre systéme peut voir des pics de 1200 lectures par secondes sans aucune ècritures, notre temps d'ècriture moyen est deux fois plus temps qu'une lecture, et la relation est linèaire. Supposons que notre maître et notre esclave sont de la même capacitè, et que nous avons N esclaves et un maître. Nous avons alors pour chaque serveur (maître ou esclave) :

`lectures = 1200 - 2 * ècriture` (issue des tests)

`lectures = 9* ècriture / (N + 1)` (lectures rèparties, mais toutes les ècritures vont á tous les serveurs)

`9*ècriture/(N+1) + 2 * ècriture = 1200`

`ècriture = 1200/(2 + 9/(N+1)`

Si N = 0, ce qui signifie que nous n'avons pas de rèplication, notre systéme peut gèrer 1200/11, environs 109 ècritures par secondes, ce qui signifie (que nous aurons 9 fois plus de lectures que d'ècritures, ètant donnè la nature de notre application).

Si N = 1, nous pouvons monter á 184 ècriture par seconde.

Si N = 8, nous pouvons monter á 400 ècriture par seconde.

Si N = 17, nous pouvons monter á 480 ècriture par seconde.

Eventuellement, si N se rapproche de l'infini (et notre budget de l'infini nègatif), nous pourrons nous rapprocher de 600 ècritures par secondes, en amèliorant le systéme 5,5 fois. Toutefois, avec 8 serveurs, nous avons pu amèliorer le systéme de 4 fois.

Notez que nos calculs ont supposès une bande passante infinie, et que nous avons nègligè des facteurs qui pourraient être significatifs pour notre systéme. Dans de nombreux cas, nous ne pourrions pas faire de calculs prècis pour prèdire l'ètat de notre systéme avec N esclaves de rèplication. Toutefois, rèpondre aux questions ci-dessus vous permettra de dècider si la rèplication est une solution á votre probléme ou pas.

- Quel est le ratio d'ècriture/lecture de votre systéme?
- Quelle est la charge maximale d'un serveur en ècriture, si vous pouvez limiter les lectures?
- Combien d'esclaves votre rèseau peut supporter?

Q : Comment puis-je utiliser la rèplication pour fournir un systéme á haute tolèrance de panne?

R : Avec les fonctionnalitès actuellement disponible, vous devez configurer un serveur et un esclave (ou plusieurs esclaves), et ècrire un script qui va surveiller le maître pour voir si il fonctionne , et instruire votre applcation et les esclaves d'un changement de maître en cas d'èchec. Voici des suggestions :

- Utilisez la commande `CHANGE MASTER TO` pour changer un esclave en maître.
- Un bon moyen de garder votre application informè du maître courant est d'utiliser les DNS dynamiques, vous pouvez attribuer au maître. Avec `bind`, vous pouvez utiliser '`nsupdate`' pour modifier dynamiquement votre DNS.
- Vous devez faire fonctionner vos esclaves avec l'option `log-bin` et sans l'option `log-slave-updates`. De cette façon, l'esclave sera prêt á prendre le relais dés que vous lui enverrez la commande `STOP SLAVE`; envoyez `RESET MASTER` et `CHANGE MASTER TO` aux autres esclaves. Cela va aussi vous aider á intercepter des modifiations contagieuses qui pourraient intervenir dûes á des mauvaise configurations d'esclaves (idèalement, vous

allez donner des droits d'accés pour que personne ne puisse modifier l'esclave, hormis le thread esclave), combinèes avec des bogues dans vos programmes clients (ils ne doivent jamais mettre á jour un esclave directement).

Nous travaillons actuellement á l'intègration automatique de l'èlection d'un nouveau maître, mais jusqu'á ce que ce soit prés, vous devez crèer votre propre outil de surveillance.

4.10.8 Correction de problémes courants

Si vous avez suivi les instructions, et que votre configuration de rèplication ne fonctionne pas, commencez par supprimer les problémes liès á l'utilisateur comme ceci :

- Est ce que le maître enregistre dans le log binaire ? Vèrifiez avec la commande `SHOW MASTER STATUS`. Si il le fait, la variable `Position` doit être non nulle. Si ce n'est pas le cas, vèrifiez que vous avez donnè au serveur l'option `log-bin` et que vous lui avez donnè un `server-id`.
- Est ce que l'esclave fonctionne? Vèrifiez le avec `SHOW SLAVE STATUS`. La rèponse se trouve dans la colonne `Slave_running`. Si ce n'est pas le cas, vèrifiez les options de l'esclave, et vèrifiez le fichier de log d'erreurs.
- Si l'esclave fonctionne, as-t-il ètablit une connexion avec le maître? Exècutez la commande `SHOW PROCESSLIST`, et recherchez un utilisateur avec la valeur `system user` dans la colonne `User` et `none` dans la colonne `Host`, et vèrifiez la colonne `State`. Si elle indique `connecting to master`, vèrifiez les droits de connexion pour l'utilisateur de rèplication sur le serveur, ainsi que le nom de l'hôte, votre configuration DNS, le fonctionnement du maître, et si tout est OK, vèrifiez le fichier de log d'erreurs.
- Si l'esclave fonctionnait, mais s'est arrêtè, vèrifiez le rèsultat de la commande SHOW SLAVE STATUS, et vèrifiez le fichier de log d'erreurs. Il arrive que certaines requêtes rèussissent sur le maître mais èchouent sur l'esclave. Cela ne devrait pas arriver si vous avez pris la bonne sauvegarde du maître, et que vous n'avez jamais modifiè les donnèes sur le serveur esclave, autrement que par le truchement de l'esclave de rèplication. Si c'est le cas, c'est un bogue, et vous devez le rapporter. Voyez plus loin pour savoir comment rapporter un bogue.
- Si une requête qui a rèussit sur le maître, refuse de s'exècuter sur l'esclave, et qu'une synchronisation compléte de la base ne semble pas possible, essayez ceci :
  - Commencez par voir si il n'y a pas de lignes dans le chemin. Essayez de comprendre comment a plus se trouver la, effacez la, et essayer de redèmarrer l'esclave avec `SLAVE START`
  - Si la solution ci-dessus ne fonctionne pas ou ne s'applique pas, essayez de comprendre si c'est risquè de faire une correction á la main (au besoin) puis, ignorez la prochaine requête du maître.
  - Si vous avez dècidè que vous pouvez vous passer de la prochaine requête, exècutez les commandes `SET GLOBAL SQL_SLAVE_SKIP_COUNTER=1; SLAVE START;` pour ignorer toutes les requêtes qui n'utilisent pas `AUTO_INCREMENT` ou `LAST_INSERT_ID()`, ou `SET GLOBAL SQL_SLAVE_SKIP_COUNTER=2; SLAVE START;`. La raison pour laquelle des requêtes qui utilisent `AUTO_INCREMENT` or `LAST_INSERT_ID()` sont diffèrentes, est que elles reprèsentent deux èvènements dans le log binaire du maître.

  - Si vous êtes sûr que l'esclave a ètè dèmarrè en parfaite synchronisation avec le maître, et que personne n'a fait de modification dans les tables impliquèes, en dehors de l'esclave, lisez les rapport de bogues, pour vous èviter d'essayer tous les trucs ci-dessus.
- Assurez vous que vous ne rencontrez pas un des vieux bogues que vous pourriez supprimer en changeant de versions.
- Si tout cela èchoue, lisez les logs d'erreurs. Si ils sont gros, utilisez la commande `grep -i slave /path/to/your-log.err` sur l'esclave. Il n'y a pas de schèma de recherche particulier a appliquer au maître, car les seules erreurs qu'il stocke sont des erreurs gènèrales : si il le peut, il envoie l'erreur á l'esclave.

Lorsque vous avez èpuisè toutes les possibilitès, et qu'il n'y a pas d'erreur utilisateur impliquèe, et que la rèplication ne fonctionne plus du tout ou qu'elle est trés instable, il est temps de prèparer un rapport de bogues. Essayer de passer un peu de temps sur ce rapport pour en faire un bon. Idèalement, vous souhaitons recevoir un cas de test au format prèsentè dans le dossier `mysql-test/t/rpl*`, du dossier source. Si vous soumettez un test comme cela, vous pouvez vous attendre á recevoir un patch dans un ou deux journèes, même si la durèe de rèaction peut varier en fonction de nombreux facteurs.

La deuxiéme meilleure option est simplement d'ècrire un programme qui va facilement configurer les arguments de connexion de votre maître et esclave, et qui va illustrer le probléme sur nos systémes. Vous pouvez l'exècuter en C ou en Perl, suivant votre langage de programmation.

Si vous pouvez utiliser l'une des mèthodes ci-dessus pour dèmontrer votre bogue, utilisez le script `mysqlbug` pour prèparer le rapport de bogue, et envoyez le á `bugs@lists.mysql.com`. Si vous avez un probléme fantôme (un bogue qui survient, mais que vous ne pouvez pas reproduire á coup sûr) :

- Vèrifiez qu'il y n'y a pas d'autre utilisateur impliquè. Par exemple, si vous modifiez l'esclave en dehors du thread l'esclave, les donnèes seront dèsynchronisèes, et vous pourriez aboutir á des violations de contraintes, auquel cas, vous devez arrêter l'esclave et nettoyer manuellement les tables pour les synchroniser.
- Exècutez l'esclave avec les options `log-slave-updates` et `log-bin` – cela va conserver un log de toutes les modifications dans le cadre de l'esclave.
- Sauvez toutes les preuves avant d'èteindre la rèplication. Si nous n'avons que des informations schèmatiques, cela nous prendra beaucoup de temps pour ètudier le probléme. Les preuves que vous devriez obtenir sont :
  - Le fichier de log binaire du maître
  - Tous les fichiers de logs binaires sur l'esclave
  - Le rèsultat de la commande `SHOW MASTER STATUS` sur le maître, au moment de la dècouverte du probléme.
  - Le rèsultat de la commande `SHOW SLAVE STATUS` sur le maître, au moment de la dècouverte du probléme.
  - Les fichiers de log d'erreurs du maître et de l'esclave.
- Utilisez `mysqlbinlog` pour examiner les logs binaires. Cette instructions doit être pratique pour rechercher des requêtes problèmatiques :

```
mysqlbinlog -j pos_from_slave_status /path/to/log_from_slave_status | head
```

Une fois que vous avez rassemblè les preuves du probléme fantôme, essayez au maximum de l'isoler dans un cas de test. Puis, rapporter le probléme á `bugs@lists.mysql.com` avec autant d'information possibles.

5 Optimisation de MySQL

L'optimisation est une tâche complexe car elle nècessite une parfaite comprèhension du systéme en entier. Alors qu'il serait possible de faire quelques optimisations localement avec une faible connaissance de votre systéme ou de votre application, plus vous voulez un systéme optimal, plus il est nècessaire de le connaître.

Ce chapitre va tenter d'expliquer et de donner des exemples de diffèrentes maniéres d'optimiser MySQL. Souvenez-vous, malgrè tout, qu'il existe toujours d'autres moyens (de plus en plus difficiles) de rendre le systéme plus vèloce.

5.1 Vue d'ensemble de l'optimisation

Le plus important pour obtenir un systéme rapide est bien sûr le schèma utilisè. Il faut ègalement savoir le genre de choses que fera le systéme et oú seront les goulots d'ètranglement.

Les ralentissements les plus courants sont :

- Les recherches sur les disques. Il faut du temps pour que le disque trouve un morceau de donnèe. Avec les disques modernes en 1999, le temps d'accés est gènèralement infèrieur á 10ms, donc on peut en thèorie faire prés de 100 recherches par seconde. Le temps s'amliore un peu avec les nouveaux disques et il est difficile de l'optimiser pour une seule table. La mèthode pour l'optimiser est de dissèminer les donnèes sur plus d'un disque.
- Les accés disques en lecture/ècriture. Une fois le disque á la bonne position, il faut lire les donnèes. Avec des disques modernes en 1999, un disque fournit environ 10 á 20Mo/s. Ceci est plus facile á optimiser que les recherches car on peut lire en paralléle sur plusieurs disques.
- Les cycles CPU. Quand les donnèes sont dans la mèmoire centrale (ou si elles y ètaient dèjá) il faut les traiter pour obtenir le rèsultat. Avoir des tables de petite taille par rapport á la mèmoire est le facteur limitant le plus courant. Mais avec des petites tables, la cèlèritè n'est souvent pas un souci.
- La bande passante de la mèmoire. Quand le CPU a besoin de plus d'informations qu'il ne peut en stocker dans son cache, la bande passante de la mèmoire centrale devient le goulot d'ètranglement. C'est un ètranglement peu courant pour la majoritè des systémes, mais il faut être averti.

5.1.1 Limitations et inconvènients des choix conceptuels de MySQL

Avec les tables de type MyISAM, MySQL utilise un verrouillage extrêmement rapide (plusieurs lectures / une seule ècriture). Le plus gros probléme avec ce type de table survient quand vous avez un mèlange de flux de modifications et des sèlections lentes sur la même table. Si c'est une probléme sur plusieurs tables, vous pouvez utiliser un autre type de table pour celles ci. Voir Chapitre 7 [Table types], page 531.

MySQL peut utiliser á la fois des tables transactionnelles et des tables non-transactionnelle. Pour pouvoir travailler tranquillement avec des tables non-transactionnelles (qui n'ont pas

la possibilitè de revenir en arriére si quelque chose se passe mal) MySQL suit les régles suivantes:

- Toutes les colonnes ont une valeur par dèfaut.
- Si vous insèrez une mauvaise valeur dans une colonne (par exemple `NULL` dans une colonne `NOT NULL`, ou encore une valeur numèrique trop grande dans une colonne numèrique), MySQL prendra en compte "la meilleure valeur possible" plutôt que de sortir une erreur. Pour les valeurs numèriques, il s'agit de 0, de la valeur la plus petite possible, ou de la valeur la plus grande possible. Pour les chaînes, il s'agit soit d'une chaîne vide, soit de la chaîne la plus longue que peut contenir la colonne.
- Toutes les expressions calculèes retournent une valeur qui peut être utilisèes á la place d'afficher un message d'erreur. Par exemple, 1/0 retourne `NULL`

La raison des régles prècitèes est que nous ne pouvons pas vèrifier les conditions avant que la requête ne commence á être exècutèe. Si nous rencontrons un probléme aprés avoir mis á jour plusieurs lignes, nous ne pouvons plus revenir en arriére puisque la table ne le supporte pas. Nous ne pouvons pas nous arrêter car le travail de mise á jour ne serait fait qu'á moitiè, ce qui est sans doute le pire qui puisse arriver. Dans ce cas, il vaut mieux faire du mieux qu'on peut et continuer comme si rien ne s'ètait passè.

Ce qui prècéde signifie qu'il ne faut pas que le contrôle du contenu des champs soit fait au niveau de MySQL, mais au niveau de l'application.

5.1.2 Portabilitè

Comme tous les serveurs SQL implèmentent diffèremment le langage SQL, cela prend de solides connaissances pour ècrire des applications SQL portables. Pour les insertions et sèlections simples, c'est trés simple, mais plus vos besoins se complexifient, plus c'est abscons. Si vous voulez une application qui fonctionne rapidement sur de nombreuses bases de donnèes, c'est même encore plus difficile.

Pour rendre une application complexe portable, vous pouvez commencer par choisir une panoplie de serveurs SQL avec lesquels travailler.

Vous pouvez utiliser le programme/page web de MySQL appelè `crash-me` `http://www.mysql.com/information/crash-me.php` pour trouver les fonctions, types et limites que vous pouvez utiliser avec un panel de serveurs de bases de donnèes. Les tests de `crash-me` ne vèrifient pas tout, mais il est dèjá trés exhaustif avec plus de 450 points de tests.

Par exemple, vous ne devriez pas avoir de nom de colonne supèrieur á 18 caractéres, si vous voulez pouvoir utiliser Informix ou DB2.

Les programmes de tests `crash-me` et de performances de MySQL sont trés indèpendants du serveur. En regardant comment nous avons gèrè ces situations, vous pouvez comprendre comment rendre votre propre code indèpendant du serveur. Les tests de performances sont situès dans le dossier '`sql-bench`' de la distribution source de MySQL. Ils sont ècrits en Perl avec l'interface DBI, ce qui rèsout les problémes de connexion.

Voyez `http://www.mysql.com/information/benchmarks.html` pour connaître les rèsultats de ces benchmarks.

Comme vous pouvez le voir avec ces rèsultats, toutes les bases de donnèes ont leur point faible. En rèalitè, elles ont toutes une approche diffèrente du même probléme, et cela conduit á des comportements spècifiques.

Si vous avez besoin de l'indèpendance au serveurs de bases de donnèes, vous devez bien connaître les faiblesses de chaque serveur. MySQL est trés rapide pour lire et modifier les donnèes, mais peine lorsque les lectures et ècritures sont lentes sur la même table. Oracle, d'un autre cotè, a de gros problémes lorsque vous essayez d'accèder aux donnèes que vous avez modifiè rècemment (jusqu'á ce qu'elles soient ècrites sur le disque). Les bases de donnèes transactionnelles en gènèral ne sont pas trés douèes pour gènèrer des tables rèsumès á partir des tables de log, car dans ce cas, le verrouillage de ligne est inutile.

Pour rendre votre application *reellement* indèpendante de la base de donnèees, vous devez dèfinir un classe trés souple á travers laquelle vous allez vous interfacer pour manipuler vos donnèes. Comme le langage C++ est disponible sur la plupart des systémes, cela rend les classes C++ trés pratiques pour cette tâche.

Si vous utilisez une fonctionnalitè spècifique d'une base de donnèes (comme la commande `REPLACE` de MySQL), il vous faut aussi coder la même commande pour les autres serveurs (qui sera alors plus lente). Avec MySQL, vous pouvez aussi utiliser la syntaxe `/*! */` pour utiliser des mots clès spècifiques de MySQL dans une requête. Le code entre `/* */` sera alors traitè comme un commentaire et ignorè par la plupart des autres serveurs SQL.

Si les hautes performances sont plus importantes que l'exactitude, comme pour les applications web, il est possible de crèer une couche application qui met en cache les rèsultats et vous donne de meilleures performances. En laissant les anciens rèsultats se pèrimer, vous pouvez garder un cache á jour. Cela vous donne une mèthode pour gèrer les grandes charges, durant lesquelles vous pouvez augmenter la taille du cache, et augmenter la durèe de vie.

Dans ce cas, les informations de crèation de tables doivent contenir les informations de taille initiale du cache, et la frèquence de rafraîchissement des tables.

5.1.3 Pour quoi avons nous utilisè MySQL ?

Pendant le dèveloppement initial de MySQL, les fonctions de MySQL ont ètè crèèes pour convenir á un maximum de clients. Celles ci supporte des entrepôts de donnèes pour deux des plus gros revendeurs suèdois.

Nous recevons chaque semaine le rèsumè de toutes les transactions par carte de toutes les boutiques, et nous sommes chargès de fournir des informations utiles aux gèrants des boutiques pour les aider á comprendre comment leurs propres campagnes publicitaires touchent leurs clients.

Les donnèes sont assez ènormes (prés de 7 millions de rèsumès de transactions par mois), et nous avec les donnèes de 4-10 ans que nous prèsentons aux utilisateurs. Nous avons chaque semaine des requêtes des clients qui veulent un accés 'instantanè' aux nouveaux rapports sur ces donnèes.

Nous avons rèussi en stockant toutes les informations dans des tables de 'transactions' compressèes. Nous avons une sèrie de macros (scripts) qui gènére des tables de rèsumès groupès par diffèrents critéres (groupe de produits, identifiant de client, boutique ...). ces rapports sont des pages web gènèrèes dynamiquement par un petit script Perl qui parcours

une page web, exècute les requêtes SQL, et insére les rèsultats. Nous aurions bien utilisè PHP ou mod_perl á la place, mais ils n'ètaient pas disponibles á cette èpoque.

Nous avons ècrit un outil en `C` pour la reprèsentation graphique des donnèes qui gènére des GIFs á partir du rèsultat de requêtes SQL (avec quelques traitements sur le rèsultat). Ceci est ègalement effectuè dynamiquement par le script Perl qui parcourt les fichiers `HTML`.

Pour la plupart des cas, un nouveau rapport peut simplement être fait en copiant un script existant, et en modifiant la requête SQL qu'il exècute. Dans certains cas, nous aurons besoin d'ajouter des champs á une table de rèsumè existante ou d'en gènèrer une nouvelle, mais c'est tout de même toujours assez simple, car nous gardons toutes les tables de transactions sur disque. (Actuellement, nous avons au moins 50 Go de tables de transactions et 200 Go d'autres donnèes sur les clients.)

Nous donnons ègalement accés aux tables de rèsumès á nos clients directement avec ODBC, de sorte que les utilisateurs avancès puissent traiter les donnèes eux-mêmes .

Nous n'avons eu aucun probléme á supporter tout cela avec une relativement modeste Sun Ultra SPARCStation (2x200 MHz). Nous avons rècemment amèliorè l'un de nos serveurs en un bi-CPU 400 MHz UltraSPARC, et nous projetons actuellement de supporter les transactions au niveau du produit, ce qui signifie un dècuplement des donnèes. Nous pensons pouvoir y arriver uniquement en ajoutant des disques supplèmentaires á nos systémes.

Nous expèrimentons aussi Intel-Linux, pour pouvoir avoir plus de puissance CPU pour moins cher. Comme nous utilisons dèsormais le format binaire portable pour les bases de donnèes (nouveautè de la version 3.23), nous utiliserons cela pour quelques parties de l'application.

Nous avons au dèpart le sentiment que Linux s'acquitera mieux des faibles et moyennes charges tandis que Solaris fonctionnera mieux sur les grosses charges á cause des I/O disques extrêmes, mais nous n'avons actuellement aucune conclusion á ce propos. Aprés quelques discussion avec un dèveloppeur du noyau Linux, un effet de bord de Linux pourrait tant de ressources aux travaux de traitement que les performances de l'interface interactive peut devenir vraiment lente. Cela fait apparaître la machine trés lente et sans rèponse lorsque de gros traitements sont en cours. Heureusement, cela sera mieux gèrè dans les futurs noyaux de Linux.

5.1.4 La suite de tests MySQL

Ceci devrait comprendre une description technique de la suite de tests de performances de MySQL (et `crash-me`), mais cette description n'est pas encore ècrite. Actuellement, vous pouvez vous faire une idèe des tests en regardant le code et les rèsultats dans le rèpertoire '`sql-bench`' dans toutes les distributions de sources de MySQL.

Cette suite de test est censèe permettre á utilisateur de comparer ce qu'une implèmentation SQL donnèe rèussi bien ou mal.

Sachez que ces tests de performances lancent en un seul thread, donc il mesure le temps minimum pour chaque opèration. Nous projetons pour le futur d'ajouter de nombreux tests multi-thread á cette suite de tests.

Par exemple, (tous ont ètè lancès sur une même machine NT 4.0)

Lecture de 2000000 lignes indexèes	**Secondes**	**Secondes**

mysql	367	249
mysql_odbc	464	
db2_odbc	1206	
informix_odbc	121126	
ms-sql_odbc	1634	
oracle_odbc	20800	
solid_odbc	877	
sybase_odbc	17614	
Insertion de lignes (350768	**Secondes**	**Secondes**
mysql	381	206
mysql_odbc	619	
db2_odbc	3460	
informix_odbc	2692	
ms-sql_odbc	4012	
oracle_odbc	11291	
solid_odbc	1801	
sybase_odbc	4802	

Le test ci-dessus a ètè exècutè avec un index de cache de 8Mo.

Nous avons rassemblè d'autres rèsultats de tests á `http://www.mysql.com/information/benchmarks.html`

Notez que Oracle n'est pas inclus dans ces tests car ils ont demandè á être retirès. Tous les tests d'Oracle doivent être faits par Oracle! Nous croyons qu cette politique va biaiser **fortement** les tests en faveur de Oracle, car les tests ci-dessus sont supposè montrer ce qu'une installation simple peut faire pour un client simple.

Pour exècuter la suite de tests, vous devez tèlècharger la distribution source de MySQL, installer les pilotes perl DBI et perl DBD pour les bases qui vous intèressent.

```
cd sql-bench
perl run-all-tests --server=#
```

oú # est un des serveurs supportès. Vous pouvez obtenir une liste de toutes les options et des serveurs supportès avec l'option de ligne de commande `run-all-tests --help`.

`crash-me` essaie de dèterminer quelles fonctionnalitès un serveur supporte, et quelles sont ses limitations. Par exemple, le test dètermine :

- Les types de colonnes supportès
- Le nombre d'index supportès
- Les fonctions supportèes
- La taille maximale d'une requête
- La taille maximale d'une colonne `VARCHAR`

Vous pouvez retrouver les rèsultats de `crash-me` sur de nombreuses bases de donnèes á `http://www.mysql.com/information/crash-me.php`.

5.1.5 Utiliser vos propres tests de performance

Vous devriez vraiment penser á prèparer des tests de performances pour votre application et base, afin d'identifier les opèrations les plus lentes. En les corrigeant (ou en remplaçant ces

opèrations des 'modules simples') vous pouvez facilement identifier les autres opèrations lentes (et ainsi de suite...). Même si la performance gènèrale de votre application est suffisante, vous devriez prèvoir oú seront les prochains freins, et dècider d'anticiper leur rèsolution, avant que vous n'ayez vraiment besoin de ces performances.

Pour avoir un exemple de programme de tests portables, voyez la suite de tests MySQL. Voir Section 5.1.4 [MySQL Benchmarks], page 360. Vous pouvez prendre n'importe quel programme de cette suite, le modifier pour l'adapter á vos besoins, et essayer diffèrentes solutions á votre probléme : il suffit de tester et d'identifier la solution la plus rapide pour vous.

Il est trés frèquent que des problémes surviennent lorsque le systéme subit une forte charge. Nous avons de nombreux clients qui nous contactent lorsqu'ils ont mis leur systéme en production, et rencontrè des problémes de charge. Pour chacun d'entre eux, les problémes ètaient des problémes simples de conceptions (les scans de tables ne sont **pas bons** sous forte charge) ou des problémes liès au systéme d'exploitation ou les librairies. La plupart auraient ètè vraiment **plus simples** á tester si le systéme n'ètait pas dèjá en production.

Pour èviter des problémes comme ceux-lá, vous devriez mettre quelques efforts dans les tests de votre application dans son ensemble, avant de la mettre dans les pires conditions. Vous pouvez utiliser le programme Super Smack pour cela, qui est disponible á `http://www.mysql.com/Downloads/super-smack/super-smack-1.0.tar.gz`. Comme son nom le suggére, il va mettre votre systéme á genoux si vous lui demandez, alors assurez vous de ne l'utiliser qu'avec votre systéme de dèveloppement.

5.2 Optimisation des `SELECT`s et autres requêtes

Premiérement, ce qui affecte toutes les requêtes : plus votre systéme de droits est compliquè, plus vous aurez des baisses de performances.

Si vous n'avez aucun `GRANT` effectuè, MySQL optimisera les vèrifications de droits. Donc, si vous avez un systéme volumineux, il serait bènèfique d'èviter les grants. Sinon les performances seront rèduites.

Si votre probléme est du á des fonctions natives de MySQL, vous pouvez toujours effectuer les tests suivants avec le client MySQL :

```
mysql> SELECT BENCHMARK(1000000,1+1);
+------------------------+
| BENCHMARK(1000000,1+1) |
+------------------------+
|                      0 |
+------------------------+
1 row in set (0.32 sec)
```

Ce qui prècéde montre que MySQL peut exècuter 1 000 000 d'additions en 0.32 secondes sur un `PentiumII 400MHz`.

Toutes les fonctions MySQL sont sensè être optimisèes, mais il peut y avoir quelques exceptions et la fonction `BENCHMARK(nombre_de_fois,expression)` est un trés bon moyen de trouver ce qui cloche dans vos requêtes.

5.2.1 Syntaxe de EXPLAIN (Obtenir des informations sur les SELECT)

```
EXPLAIN nom_de_table
ou  EXPLAIN SELECT select_options
```

`EXPLAIN nom_de_table` est un synonyme de `DESCRIBE nom_de_table` ou `SHOW COLUMNS FROM nom_de_table`.

Lorsque vous ajoutez au dèbut d'une commande `SELECT` le mot clè `EXPLAIN`, MySQL va expliquer le processus qu'il va suivre pour exècuter la commande `SELECT`, et donner des dètails sur la façon avec laquelle il va joindre les tables, et dans quel ordre.

Avec l'aide de `EXPLAIN`, vous pouvez identifier les index á ajouter pour accèlèrer les commandes `SELECT`.

Vous devriez souvent utiliser la commande `ANALYZE TABLE` pour mettre á jour les statistiques de cardinalitè de vos tables, qui affectent les choix de l'optimisateur. Voir Section 4.5.2 [ANALYZE TABLE], page 264.

Vous pouvez aussi voir si l'optimisateur fait les jointures dans un ordre vraiment optimal. Pour forcer l'optimisateur á utiliser un ordre spècifique de jointure dans une commande `SELECT`, ajoutez l'attribut `STRAIGHT_JOIN` á la clause.

Pour les jointures complexes, `EXPLAIN` retourne une ligne d'information pour chaque table utilisèe dans la commande `SELECT`. Les tables sont listèes dans l'ordre dans lequel elles seront lues. MySQL rèsout toutes les jointures avec une seule passe multi-jointure. Cela signifie que MySQL lit une ligne dans la premiére table, puis recherche les lignes qui correspondent dans la seconde, puis dans la troisiéme, etc. Lorsque toutes les tables ont ètè traitèes, MySQL affiche les colonnes demandèes, et il remonte dans les tables jusqu'á la derniére qui avait encore des lignes á traiter. La prochaine ligne est alors traitèe de la même façon.

Avec MySQL version 4.1 l'affichage de `EXPLAIN` a ètè modifiè pour mieux fonctionner avec les structures comme `UNION`, sous-requêtes, et tables dèrivèes. La plus importante èvolution est l'addition de deux nouvelles colonnes : `id` et `select_type`.

Le rèsultat de la commande `EXPLAIN` est constituè des colonnes suivantes :

`id`
: identifiant de `SELECT`, le numèro sèquentiel de cette commande `SELECT` dans la requête.

`select_type`
: Type de clause `SELECT`, qui peut être :

  `SIMPLE`
  : Simple `SELECT` (sans `UNION`s ou sous-requêtes).

  `PRIMARY`
  : `SELECT` extèrieur.

  `UNION`
  : Second et autres `UNION SELECT`s.

  `DEPENDENT UNION`
  : Second et autres `UNION SELECTS`s, dèpend de la commande extèrieure.

  `SUBSELECT`
  : Premier `SELECT` de la sous-requête.

  `DEPENDENT SUBSELECT`
  : Premier `SELECT`, dèpendant de la requête extèrieure.

`DERIVED` Table dèrivèe `SELECT`.

`table` La table á laquelle la ligne fait rèfèrence.

`type` Le type de jointure. Les diffèrents types de jointures sont les suivants, dans l'ordre du plus efficace au plus lent :

`system` La table a une seule ligne (c'est une table systéme). C'est un cas spècial du type de jointure `const`.

`const` La table a au plus une ligne correspondante, qui sera lue dés le dèbut de la requête. Comme il n'y a qu'une seule ligne, les valeurs des colonnes de cette ligne peuvent être considèrèes comme des constantes pour le reste de l'optimisateur. Les tables `const` sont trés rapides, car elles ne sont lues qu'une fois.

`eq_ref` Une ligne de cette table sera lue pour chaque combinaison de ligne des tables prècèdentes. C'est le meilleur type de jointure possible, á l'exception des prècèdents. Il est utilisè lorsque toutes les parties d'un index sont utilisèes par la jointure, et que l'index est `UNIQUE` ou `PRIMARY KEY`.

`ref` Toutes les lignes avec des valeurs d'index correspondantes seront lues dans cette table, pour chaque combinaison des lignes prècèdentes. `ref` est utilisè si la jointure n'utilise que le prèfixe de gauche de la clè, ou si la clè n'est pas `UNIQUE` ou `PRIMARY KEY` (en d'autres termes, si la jointure ne peut pas sèlectionner qu'une seule ligne en fonction de la clè). Si la clè qui est utilisèe n'identifie que quelques lignes á chaque fois, la jointure est bonne.

`range` Seule les lignes qui sont dans un intervalle donnè seront lues, en utilisant l'index pour sèlectionner les lignes. La colonne `key` indique quel est l'index utilisè. `key_len` contient la taille de la partie de la clè qui est utilisèe. La colonne `ref` contiendra la valeur `NULL` pour ce type.

`index` C'est la même chose que `ALL`, sauf que seul l'arbre d'index sera lu et scannè. C'est gènèralement plus rapide que `ALL`, car le fichier d'index est gènèralement plus petit que le fichier de donnèes.

`ALL` Une analyse compléte de la table sera faîte pour chaque combinaison de lignes issue des premiéres tables. Ce n'est pas bon si la premiére table n'est pas une jointure de type `const` et c'est **trés** mauvais dans les autres cas. Normalement vous pouvez èviter ces situations de `ALL` en ajoutant des index basèe sur des parties de colonnes.

`possible_keys`

La colonne `possible_keys` indique quels index MySQL va pouvoir utiliser pour trouver les lignes dans cette table. Notez que cette colonne est totalement dèpendante de l'ordre des tables. Cela signifie que certaines clès de la colonne `possible_keys` pourraient ne pas être utilisèes dans d'autres cas d'ordre de tables.

Si cette colonne est vide, il n'y a pas d'index pertinent. Dans ce cas, vous pourrez amèliorer les performances en examinant votre clause `WHERE` pour voir si des colonnes sont susceptibles d'être indexèe. Si c'est le cas, crèez un index ad hoc, et examinez le rèsultat avec la commande `EXPLAIN`. Voir Section 6.5.4 [ALTER TABLE], page 512.

Pour connaître tous les index d'une table, utilisez le code `SHOW INDEX FROM nom_de_table`.

`key` La colonne `key` indique l'index que MySQL va dècider d'utiliser. Si la clè vaut `NULL`, aucun index n'a ètè choisi. Pour forcer MySQL á utiliser un index listè dans la colonne `possible_keys`, utilisez `USE KEY/IGNORE KEY` dans votre requête. Voir Section 6.4.1 [SELECT], page 481.

De plus, exècuter `myisamchk --analyze` (voir Section 4.4.6.1 [myisamchk syntax], page 247) ou `ANALYZE TABLE` (voir Section 4.5.2 [ANALYZE TABLE], page 264) sur la table va aider l'optimiseur á choisir les index.

`key_len` La colonne `key_len` indique la taille de la clè que MySQL a dècidè d'utiliser. La taille est `NULL` si la colonne `key` vaut `NULL`. Notez que cela vous indique combien de partie d'une clè multiple MySQL va rèellement utiliser.

`ref` La colonne `ref` indique quelle colonne ou quelles constantes sont utilisèes avec la clè `key`, pour sèlectionner les lignes de la table.

`rows` La colonne `rows` indique le nombre de ligne que MySQL estime devoir examiner pour exècuter la requête.

`Extra` Cette colonne contient des informations additionnelle sur comment MySQL va rèsoudre la requête. Voici une explication des diffèrentes chaînes que vous pourriez trouver dans cette colonne :

`Distinct` MySQL ne va pas continuer á chercher d'autres lignes que la ligne courante, aprés en avoir trouvè une.

`Not exists`
MySQL a ètè capable d'appliquer une optimisation de type `LEFT JOIN` sur la requête, et ne va pas examiner d'autres lignes de cette table pour la combinaison de lignes prècèdentes, une fois qu'il a trouvè une ligne qui satisfait le critére de `LEFT JOIN`.

Voici un exemple de cela :

```
SELECT * FROM t1 LEFT JOIN t2 ON t1.id=t2.id WHERE t2.id IS NUL
```

Supposons que `t2.id` est dèfini comme `NOT NULL`. Dans ce cas, MySQL va scanner `t1` et rechercher des lignes dans `t2` via `t1.id`. Si MySQL trouve une ligne dans `t2`, il sait que `t2.id` ne peut pas être `NULL`, et il ne va pas scanner le reste des lignes de `t2` qui ont le même `id`. En d'autres termes, pour chaque ligne de `t1`, MySQL n'a besoin que de faire une recherche dans `t2`, indèpendamment du nombre de lignes qui sont trouvèes dans `t2`.

`range checked for each record (index map: #)`
MySQL n'a pas trouvè d'index satisfaisant á utiliser. Il va, á la place, pour chaque combinaison de lignes des tables prècèdentes,

faire une vèrification de quel index utiliser (si il en existe), et utiliser cet index pour continuer la recherche. Ce n'est pas trés rapide, mais c'est plus rapide que de faire une recherche sans aucun index.

`Using filesort`
: MySQL va avoir besoin d'un autre passage pour lire les lignes dans l'ordre. Le tri est fait en passant en revue toutes les lignes, suivant le `type de jointure` est stocker la clè de tri et le pointeur de la ligne pour chaque ligne qui satisfont la clause `WHERE`. Alors, les clès sont triées. Finalement, les lignes sont triées dans l'ordre.

`Using index`
: Les informations de la colonne sont lues de la table, en utilisant uniquement les informations contenues dans l'index, sans avoir á faire d'autres lectures. Cela peut arriver lorsque toutes les colonnes utilisèes dans une table font partie de l'index.

`Using temporary`
: Pour rèsoudre la requête, MySQL va avoir besoin de crèer une table temporaire pour contenir le rèsultat. C'est typiquement ce qui arrive si vous utilisez une clause `ORDER BY` sur une colonne diffèrente de celles qui font partie de `GROUP BY`.

`Using where`
: Une clause `WHERE` sera utilisèe pour restreindre les lignes qui seront trouvèes dans la table suivante, ou envoyèe au client. Si vous n'avez pas cette information, et que la table est de type `ALL` ou `index`, vous avez un probléme dans votre requête (si vous ne vous attendiez pas á tester toutes les lignes de la table).

Si vous voulez rendre vos requêtes aussi rapide que possible, vous devriez examiner les lignes qui utilisent `Using filesort` et `Using temporary`.

Vous pouvez obtenir une bonne indication de la qualitè de votre jointure en multipliant toutes les valeurs de la colonne `rows` dans la table de la commande `EXPLAIN`. Cela est une estimation du nombre de lignes que MySQL va examiner pour exècuter cette requête. C'est aussi ce nombre qui sera utilisè pour interrompre votre requête, grâce á la variable `max_join_size`. Voir Section 5.5.2 [Server parameters], page 390.

L'exemple ci-dessous illustre comme une requête `JOIN` peut être optimisèe avec les rèsultats de la commande `EXPLAIN`.

Supposons que vous avez la requête `SELECT` suivante, et que vous l'examinez avec `EXPLAIN`:

```
EXPLAIN SELECT tt.TicketNumber, tt.TimeIn,
            tt.ProjectReference, tt.EstimatedShipDate,
            tt.ActualShipDate, tt.ClientID,
            tt.ServiceCodes, tt.RepetitiveID,
            tt.CurrentProcess, tt.CurrentDPPerson,
            tt.RecordVolume, tt.DPPrinted, et.COUNTRY,
            et_1.COUNTRY, do.CUSTNAME
        FROM tt, et, et AS et_1, do
        WHERE tt.SubmitTime IS NULL
```

```
AND tt.ActualPC = et.EMPLOYID
AND tt.AssignedPC = et_1.EMPLOYID
AND tt.ClientID = do.CUSTNMBR;
```

Pour cette exemple, nous supposons que :

- Les colonnes utilisèes sont dèclarèes comme ceci :

Table	Colonne	Type de colonne
tt	ActualPC	CHAR(10)
tt	AssignedPC	CHAR(10)
tt	ClientID	CHAR(10)
et	EMPLOYID	CHAR(15)
do	CUSTNMBR	CHAR(15)

- Les tables ont les index suivants :

Table	Index
tt	ActualPC
tt	AssignedPC
tt	ClientID
et	EMPLOYID (primary key)
do	CUSTNMBR (primary key)

- Les valeurs de `tt.ActualPC` ne sont pas rèparties ègalement.

Initialement, avant toute optimisation, la commande `EXPLAIN` produit les informations suivantes :

```
table type possible_keys                key  key_len ref  rows  Extra
et    ALL  PRIMARY                      NULL NULL    NULL 74
do    ALL  PRIMARY                      NULL NULL    NULL 2135
et_1  ALL  PRIMARY                      NULL NULL    NULL 74
tt    ALL  AssignedPC,ClientID,ActualPC NULL NULL    NULL 3872
      range checked for each record (key map: 35)
```

Comme le type `type` vaut `ALL` pour chaque table, le rèsultat indique que MySQL fait un scan complet de toutes les tables. Cela va prendre un trés long temps de calcul, car le nombre de lignes á examiner de cette façon est le produit du nombre de lignes de toutes les tables : dans notre cas, cela vaut `74 * 2135 * 74 * 3872 = 45,268,558,720` lignes. Si les tables ètaient plus grandes, cela serait encore pire.

Le premier probléme que vous avons ici, est que MySQL ne peut pas (encore) utiliser d'index sur les colonnes, si elles sont dèclarèes diffèremment. Dans ce contexte, les colonnes `VARCHAR` et `CHAR` sont les mêmes, mais elles ont ètè dèclarèe avec des tailles diffèrentes. Comme `tt.ActualPC` est dèclarèe comme `CHAR(10)` et que `et.EMPLOYID` est dèclarè comme `CHAR(15)`, il y a un probléme de taille.

Pour corriger cette disparitè, utilisez la commande `ALTER TABLE` pour agrandir la colonne `ActualPC` de 10 caractéres á 15 :

```
mysql> ALTER TABLE tt MODIFY ActualPC VARCHAR(15);
```

Maintenant, `tt.ActualPC` et `et.EMPLOYID` sont tous les deux des colonnes de type `VARCHAR(15)`. Exècuter la commande `EXPLAIN` produit maintenant le rèsultat suivant :

```
table type   possible_keys   key     key_len ref         rows    Extra
tt    ALL    AssignedPC,ClientID,ActualPC NULL NULL NULL 3872    Using where
do    ALL    PRIMARY         NULL    NULL    NULL        2135
      range checked for each record (key map: 1)
et_1  ALL    PRIMARY         NULL    NULL    NULL        74
      range checked for each record (key map: 1)
et    eq_ref PRIMARY         PRIMARY 15      tt.ActualPC 1
```

Ce n'est pas parfait, mais c'est bien mieux. Le produit de toutes les lignes a ètè divisè par 74). Cette version s'exècute en quelques secondes.

Une autre modification peut être faîte pour èliminer les problémes de taille de colonne pour `tt.AssignedPC = et_1.EMPLOYID` et `tt.ClientID = do.CUSTNMBR` :

```
mysql> ALTER TABLE tt MODIFY AssignedPC VARCHAR(15),
    ->                MODIFY ClientID   VARCHAR(15);
```

Maintenant, `EXPLAIN` produit le rèsultat suivant :

```
table type   possible_keys   key      key_len ref           rows Extra
et    ALL    PRIMARY         NULL     NULL    NULL          74
tt    ref    AssignedPC,     ActualPC 15      et.EMPLOYID   52   where used
             ClientID,
             ActualPC
et_1  eq_ref PRIMARY         PRIMARY  15      tt.AssignedPC 1
do    eq_ref PRIMARY         PRIMARY  15      tt.ClientID   1
```

C'est presque aussi bon que cela pourrait l'être.

Le probléme final est que, par dèfaut, MySQL support que les valeurs de la colonne `tt.ActualPC` sont uniformèment rèpartie, et que ce n'est pas le cas pour la table `tt`. Mais il est facile de le dire á MySQL :

```
shell> myisamchk --analyze PATH_TO_MYSQL_DATABASE/tt
shell> mysqladmin refresh
```

Maintenant, la jointure est parfaite, et la commande `EXPLAIN` produit ce rèsultat :

```
table type   possible_keys key     key_len ref           rows Extra
tt    ALL    AssignedPC    NULL    NULL    NULL          3872 where used
             ClientID,
             ActualPC
et    eq_ref PRIMARY       PRIMARY 15      tt.ActualPC   1
et_1  eq_ref PRIMARY       PRIMARY 15      tt.AssignedPC 1
do    eq_ref PRIMARY       PRIMARY 15      tt.ClientID   1
```

Notez que la colonne `rows` dans le rèsultat de `EXPLAIN` est une prèdiction èclairèe de l'optimisateur de jointure MySQL. Pour optimiser une requête, vous devriez vèrifier si ces nombres sont proches de la rèalitè. Si ce n'est pas le cas, vous pourriez obtenir de meilleures performances avec l'attribut `STRAIGHT_JOIN` dans votre commande `SELECT`, et en choisissant vous même l'ordre de jointure des tables dans la clause `FROM`.

5.2.2 Mesurer les performances d'une requête

Dans la plupart des cas, vous pouvez mesurer la performance d'une requête en comptant le nombre d'accés disques. Pour les tables de petite taille, vous pouvez gènèralement obtenir

une seule lecture (car l'index est probablement en cache). Pour les tables plus grandes, vous pouvez estimer que vous aurez besoin de (en utilisant les index B tree) : `log(row_count) / log(index_block_length / 3 * 2 / (index_length + data_pointer_length)) + 1` lectures pour trouver une ligne.

Pour MySQL, un bloc d'index vaut gènèralement 1024 octets, et le pointeur de donnèes vaut 4 octets. Une table de 500,000 avec un index de taille 3 (entier moyen) vous donnera : `log(500,000)/log(1024/3*2/(3+4)) + 1` = 4 lectures.

Comme l'index ci-dessus vous serait de taille 500,000 * 7 * 3/2 = 5.2Mo, (en supposant que les index des tampons sont remplit aux 2/3, ce qui est typique), vous aurez probablement l'essentiel de l'index en mèmoire, et vous n'aurez alors besoin que de 1 ou 2 lectures pour lire le reste des lignes.

Pour les ècritures, toutefois, vous aurez besoin de 4 lectures (comme ci-dessus), pour trouver la place du nouvel index, et normalement, deux autres lectures pour modifier l'index et la ligne.

Notez que le raisonnement ci-dessus n'indique pas que votre application va dègènèrer en fonction du log N ! Tant que toute est mis en cache par l'OS ou le serveur SQL, les performances ne vont se rèduire que marginalement, même si la table grossit beaucoup. Une fois que les donnèes seront trop grosses pour être en cache, votre application va ralentir car le serveur devra faire des lectures sur le disque (ce qui va accroître le log N). Pour èviter cela, augmentez le cache d'index au fur et á mesure que votre index grossit. Voir Section 5.5.2 [Server parameters], page 390.

5.2.3 Vitesse des requêtes `SELECT`

En gènèral, lorsque vous voulez rendre un `SELECT ... WHERE` plus rapide, la premiére chose á faire est de voir si vous pouvez ajouter des index. Voir Section 5.4.3 [MySQL indexes], page 384. Toutes les rèfèrences entre les tables doivent normalement être faites avec des index. Vous pouvez utiliser la commande `EXPLAIN` pour dèterminer les index utilisès pour le `SELECT`. Voir Section 5.2.1 [`EXPLAIN`], page 363.

Quelques conseils gènèraux :

- Pour aider MySQL á mieux optimiser les requêtes, exècutez `myisamchk --analyze` sur une table aprés l'avoir remplie avec quelques donnèes consistantes. Cela met á jour une valeur pour chaque partie de l'index qui indique le nombre moyen de lignes qui ont la même valeur. (Pour les index uniques, c'est toujours 1, bien sûr.) MySQL utilisera cela pour dècider quel index choisir pour connecter deux tables avec une 'expression non-constante'. Vous pouvez vèrifier le retour de l'exècution d'`analyze` en faisant `SHOW INDEX FROM nom_de_table` et examiner la colonne `Cardinality`.
- Pour trier un index et des donnèes par rapport á un index, utilisez `myisamchk --sort-index --sort-records=1` (si vous voulez trier selon le premier index). Si vous avec un index unique á partir duquel vous voulez lire toutes les lignes en prenant comme ordre cet index, c'est un bon moyen de rendre les traitements plus rapides. Notez, toutefois, que ce tri n'est pas le plus optimal et prendra beaucoup de temps pour une grosse table !

5.2.4 Comment MySQL optimise les clauses WHERE

Les optimisation de la clause WHERE sont prèsentèes avec la commande SELECT car elles sont gènèralemnt utilisèes avec la commande SELECT, mais les mêmes optimisations peuvent s'appliquer aux clauses WHERE des commandes DELETE et UPDATE.

Notez aussi que cette section est incompléte. MySQL fait de trés nombreuses optimisations, et nous n'avons pas eu le temps de toutes les documenter.

Certaines des optimisations effectuèes par MYSQL sont prèsentèes ici :

- Suppression des parenthéses inutiles :

```
   ((a AND b) AND c OR (((a AND b) AND (c AND d))))
-> (a AND b AND c) OR (a AND b AND c AND d)
```

- Remplacement des constantes :

```
   (a<b AND b=c) AND a=5
-> b>5 AND b=c AND a=5
```

- Suppression des conditions constantes (nècessaires pour le remplacement des constantes) :

```
   (B>=5 AND B=5) OR (B=6 AND 5=5) OR (B=7 AND 5=6)
-> B=5 OR B=6
```

- Les expressions constantes utilisèes par les index sont èvaluèes une fois.
- COUNT(*) sur une table simple, sans clause WHERE est lu directement dans les informations de la table pour les tables MyISAM et HEAP. Cela peut aussi être fait avec les expressions NOT NULL lorsqu'elles sont utilisèes sur une seule table. table.
- Dètection prècoce est expressions constantes invalides. MySQL dètecte rapidement les commandes SELECT qui sont impossibles, et ne retourne aucune ligne.
- HAVING est combinè avec la clause WHERE si vous n'utilisez pas la clause GROUP BY ou les fonctions de groupe (COUNT(), MIN()...).
- Pour chaque sous-jointure, une clause WHERE simplifièe est construite pour accèlèrer l'èvaluation de WHERE pour chaque sous-joinutre, et aussi essayer d'ignorer les lignes le plus tôt possible.
- Toutes les tables constantes sont lues en premier, avant toute autre table de la requête. Une table constante est une table :
  - Une table vide ou une table d'une ligne.
  - Une table qui est utilisèe avec la clause WHERE sur un index de type UNIQUE, ou avec une clè primaire PRIMARY KEY, dont toutes les parties sont des expressions constantes, et les parties de l'index sont identifièes comme NOT NULL.

  Toutes les tables suivantes sont considèrèes comme constantes :

```
mysql> SELECT * FROM t WHERE primary_key=1;
mysql> SELECT * FROM t1,t2
    ->           WHERE t1.primary_key=1 AND t2.primary_key=t1.id;
```

- La meilleure combinaison de jointure est obtenue en testant toutes les possibilitès. Si toutes les colonnes des clauses ORDER BY et GROUP BY proviennent de la même table, cette table sera utilisèe de prèfèrence comme premiére table dans la jointure.

- Si il ya une clause `ORDER BY` et une clause `GROUP BY` diffèrente, ou si la clause `ORDER BY` ou `GROUP BY` contient des colonnes issues des tables autres que la premiére, une table temporaire est crèèe.
- Si vous utilisez `SQL_SMALL_RESULT`, MySQL va utiliser une table temporaire en mèmoire.
- Chaque index de table est interrogè, et le meilleur index qui reprèsente moins de 30% des lignes est utilisè. Si un tel index ne peut être identifiè, un scan rapide de la table est fait.
- Dans certains cas, MySQL peut lire des lignes depuis l'index sans même consulter le fichie de donnèes. Si toutes les colonnes de l'index sont des nombres, alors seul l'arbre d'index sera utilisè pour rèsoudre la requête.
- Avant chaque affichage de ligne, celles qui ne satisfont pas les critéres de la clause `HAVING` sont ignorèes.

Quelques exemples de requêtes trés rapides :

```
mysql> SELECT COUNT(*) FROM tbl_name;
mysql> SELECT MIN(key_part1),MAX(key_part1) FROM tbl_name;
mysql> SELECT MAX(key_part2) FROM tbl_name
    ->        WHERE key_part_1=constant;
mysql> SELECT ... FROM tbl_name
    ->        ORDER BY key_part1,key_part2,... LIMIT 10;
mysql> SELECT ... FROM tbl_name
    ->        ORDER BY key_part1 DESC,key_part2 DESC,... LIMIT 10;
```

Les requêtes suivantes ne sont rèsolues qu'avec l'arbre d'index (en supposant que les colonnes sont numèriques) :

```
mysql> SELECT key_part1,key_part2 FROM tbl_name WHERE key_part1=val;
mysql> SELECT COUNT(*) FROM tbl_name
    ->        WHERE key_part1=val1 AND key_part2=val2;
mysql> SELECT key_part2 FROM tbl_name GROUP BY key_part1;
```

Les requêtes suivantes utilisent l'indexation pour lire les lignes dans un ordre donnès, dans faire de tri supplèmentaire :

```
mysql> SELECT ... FROM tbl_name
    ->            ORDER BY key_part1,key_part2,... ;
mysql> SELECT ... FROM tbl_name
    ->            ORDER BY key_part1 DESC,key_part2 DESC,... ;
```

5.2.5 Comment MySQL optimise la clause `DISTINCT`

`DISTINCT` est converti en `GROUP BY` sur toutes les colonnes, `DISTINCT` combinè avec un `ORDER BY` aura dans la plupart des cas recours á une table temporaire.

Quand vous combinerez `LIMIT #` avec `DISTINCT`, MySQL stoppera dés qu'il trouvera `#` lignes uniques.

Si vous n'utilisez pas de colonnes de toutes les tables utilisèes, MySQL arrête de scanner la table non-utilisèe dés qu'il trouve la premiére correspondance.

```
SELECT DISTINCT t1.a FROM t1,t2 where t1.a=t2.a;
```

Dans ce cas, en supposant que `t1` est utilisèe avant `t2` (vèrifiez agvec `EXPLAIN`), MySQL arrêtera de lire á partir de `t2` (pour cette ligne particuliére de `t1`) lorsque la permiére ligne de `t2` est trouvèe.

5.2.6 Comment MySQL optimise `LEFT JOIN` et `RIGHT JOIN`

`A LEFT JOIN B` est implèmentè dans MySQL comme suit :

- La table `B` est censèe être dèpendante de la table `A` et de toutes les tables dont dèpend `A`.
- La table `A` est censèe être dèpendante de toutes les tables (á part `B`) qui sont utilisèes dans la condition du `LEFT JOIN`.
- Toutes les conditions du `LEFT JOIN` sont transmises á la clause `WHERE`.
- Toutes les optimisations standards de jointures sont effectuèes, á l'exceptè qu'une table est toujours lue aprés celles dont elle dèpend. S'il y'a une dèpendance circulaire, MySQL retournera une erreur.
- Toutes les optimisations standards de `WHERE` sont effectuèes.
- S'il y'a une ligne dans `A` qui rèpond á la clause `WHERE`, mais qu'il n'y avait aucune ligne dans `B` qui rèpondait á la condition du `LEFT JOIN`, alors une ligne supplèmentaire de `B` est gènèrèe avec toutes les colonnes mises á `NULL`.
- Si vous utilisez `LEFT JOIN` pour trouver les enregistrements qui n'existent pas dans d'autres tables et que vous effectuez le test suivant : `nom_colonne IS NULL` dans la partie `WHERE`, oú `nom_colonne` est une colonne qui est dèclarèe en tant que `NOT NULL`, alors MySQL arrêtera de chercher d'autres lignes (pour une combinaison de clefs particuliére) aprés avoir trouvè une ligne qui rèpond á la condition du `LEFT JOIN`.

`RIGHT JOIN` est implèmentè de maniére analogue á `LEFT JOIN`.

L'ordre de lecture de tables forcè par `LEFT JOIN` et `STRAIGHT JOIN` aidera l'optimiseur de jointures (qui calcule l'ordre dans lequel les tables doivent être jointes) á faire son travail plus rapidement, puisqu'il y'aura moins de permutations de tables á vèrifier.

Notez que ce qui prècéde signifie que si vous faites une requête de la sorte :

```
SELECT * FROM a,b LEFT JOIN c ON (c.key=a.key) LEFT JOIN d (d.key=a.key)
         WHERE b.key=d.key
```

MySQL fera une recherche compléte sur `b` puisque le `LEFT JOIN` forcera sa lecture avant celle de `d`. Un palliatif est de changer la requête en :

```
SELECT * FROM b,a LEFT JOIN c ON (c.key=a.key) LEFT JOIN d (d.key=a.key)
         WHERE b.key=d.key
```

5.2.7 Comment MySQL optimise les clauses `ORDER BY`

Dans certain cas, MySQL peut utiliser un index pour rèpondre á une requête `ORDER BY` ou `GROUP BY` sans faire aucun tri.

L'index peut être utilisè même si le `ORDER BY` ne correpond pas exactement á l'index, tant que toutes les parties inutilisèe de l'index et les colonnes du `ORDER BY` sont constantes dans

la clause WHERE. Les requêtes suivantes utilisent l'index pour rèpondre aux parties ORDER BY / GROUP BY :

```
SELECT * FROM t1 ORDER BY partie_clef1,partie_clef2,...
SELECT * FROM t1 WHERE partie_clef1=constante ORDER BY partie_clef2
SELECT * FROM t1 WHERE partie_clef1=constante GROUP BY partie_clef2
SELECT * FROM t1 ORDER BY partie_clef1 DESC,partie_clef2 DESC
SELECT * FROM t1 WHERE partie_clef1=1 ORDER BY partie_clef1 DESC,partie_clef2 DESC
```

Quelques cas oú MySQL ne peut **pas** utiliser les index pour rèpondre á ORDER BY: (Notez que MySQL utilisera quand même les indexes pour trouver les lignes qui correspondent á la clause WHERE) :

- Vous effectuez un ORDER BY sur des clefs diffèrentes :

  ```
  SELECT * FROM t1 ORDER BY key1,key2
  ```

- Vous effectuez un ORDER BY en utilisant des parties de clef non consècutives.

  ```
  SELECT * FROM t1 WHERE key2=constante ORDER BY partie_clef2
  ```

- Vous mèlangez ASC et DESC.

  ```
  SELECT * FROM t1 ORDER BY partie_clef1 DESC,partie_clef2 ASC
  ```

- La clef utilisèe pour extraire les rèsultats n'est pas la même que celle utilisèe lors du groupement ORDER BY :

  ```
  SELECT * FROM t1 WHERE key2=constant ORDER BY key1
  ```

- Vous faites une jointure entre plusieurs tables et les colonnes sur lesquelles vous faites un ORDER BY ne font pas toutes parties de la premiére table non-const qui est utilisèe pour rècupèrer les lignes (C'est la premiére table dans l'affichage d'EXPLAIN qui n'utilise pas une mèthode de rècupèration sur une ligne constante).
- Vous avez plusieurs expressions ORDER BY et GROUP BY.
- L'index de table utilisè est un type d'index qui n'enregistre pas les lignes dans l'ordre. (comme le type d'index HASH dans les tables HEAP).
- Les colonnes index peuvent contenir des valeurs NULL et l'une d'elles utilise ORDER BY ... DESC. Cela vient du fait que en SQL, les valeurs NULL sont toujours triées avant les valeurs normales, que vous utilisiez DESC ou non.

Dans les cas oú MySQL doit trier les rèsultats, il utilisera l'algorithme suivant :

- Read all rows according to key or by table scanning. Les lignes qui ne rèpondent pas á la clause WHERE sont èvitèes.
- Store the sort-key in a buffer (of size sort_buffer).
- Lorsque le tampon est plein, exècuter un qsort dessus et enregistrer le rèsultat dans un fichier temporaire. Sauvegarder un pointeur sur le block enregistrè. (Dans le cas oú toutes les lignes rentre dans le tampon de tri, aucun fichier temporaire n'est crèe)
- Rèpeter ce qui prècéde jusqu'á ce que toutes les lignes soient lues.
- Do a multi-merge of up to MERGEBUFF (7) regions to one block in another temporary file. Repeat until all blocks from the first file are in the second file.
- Rèpeter ce qui prècéde jusqu'á ce qu'il y ait moins de MERGEBUFF2 (15) blocks á traiter.
- On the last multi-merge, only the pointer to the row (last part of the sort-key) is written to a result file.

- Now the code in '`sql/records.cc`' will be used to read through them in sorted order by using the row pointers in the result file. To optimise this, we read in a big block of row pointers, sort these and then we read the rows in the sorted order into a row buffer (`record_rnd_buffer`) .

Vous pouvez vèrifier avec `EXPLAIN SELECT ... ORDER BY` si MySQL peut utiliser des index pour rèpondre á cette requête. Si vous obtenez un `Using filesort` dans la colonne `extra`, c'est que MySQL ne peut utiliser d'index pour rèsoudre cet `ORDER BY`. Voir Section 5.2.1 [EXPLAIN], page 363.

Si vous voulez plus de rapiditè avec les `ORDER BY`, vous devez d'abord voir si vous pouvez faire en sorte que MySQL utilises des index au lieu de passer par des phases de tri en plus. Si cela se rèvéle impossible, vous pouvez :

- Augmenter la taille de la variable `sort_buffer`.
- Augmenter la taille de la variable `record_rnd_buffer`.
- Changer `tmpdir` pour qu'il pointe vers un disque dèdiè avec beaucoup d'espaces libres.

5.2.8 Comment MySQL optimise la clause `LIMIT`

Dans certains cas, MySQL va gèrer la requête diffèremment avec la clause `LIMIT #`, si la clause `HAVING` n'est pas utilisèe :

- Si vous ne sèlectionnez que quelques lignes avec `LIMIT`, MySQL va utiliser les index dans certains cas, oú il aurait prèfèrè utiliser un scan de table complet.
- Si vous utilisez `LIMIT #` avec la clause `ORDER BY`, MySQL va arrêter de trier dés qu'il a trouvè la premiére `#` au lieu de trier toute la table.
- Lorsque vous combinez `LIMIT #` avec `DISTINCT`, MySQL va s'arrêter dés qu'il a trouvè `#` lignes distinctes.
- Dans certains cas, la clause `GROUP BY` peut être appliquèe en lisant les clès dans l'ordre (ou en faisant un tri sur la clè), puis en calculant un sommaire, jusqu'á ce que la clè soient modifièe. Dans ce cas, `LIMIT #` ne va pas appliquer les èlèments non nècessaires de la clause `GROUP BY`s.
- Aussitôt que MySQL a envoyè les premiéres `#` lignes au client, il annule le reste de la requête (si vous n'utilisez pas la fonction `SQL_CALC_FOUND_ROWS`).
- `LIMIT 0` va toujours retourner rapidement un rèsultat vide. C'est pratique pour vèrifier une requête et lire les types de colonnes du rèsultat, sans exècuter rèellement la requête.
- Lorsque le serveur utilise des tables temporaire pour rèsoudre les requêtes, la clause `LIMIT #` est utilisèe pour calculer l'espace nècessaire.

5.2.9 Vitesse des requêtes `INSERT`

Le temps d'insertion d'une ligne est constituè comme ceci :

- Connexion : (3)
- Envoi au serveur : (2)
- Analyse de la requête : (2)
- Insertion de la ligne : (1 x taille de la ligne)

- Insertion des index : (1 x nombre d'index)
- Fermeture : (1)

oú les nombres reprèsentent une partie proportionnelle du temps total. Le calcul ne prend pas en compte les coûts d'administration initiaux de l'ouverture des tables (qui est fait une fois pour chaque requête simultanèe).

La taille de la table ralentit les opèrations d'insertion des index par un facteur de log N (B-trees).

Quelques mèthodes pour accèlèrer les insertions :

- Si vous insèrerz plusieurs lignes depuis le même client, en même temps, utilisez les valeurs multiples de la commande `INSERT`. C'est bien plus rapide (et parfois beaucoup plus rapide) que d'utiliser des commandes `INSERT` distinctes. Si vous ajoutez des donnèes dans une table non vide, vous pouvez ajuster la variable `bulk_insert_buffer_size` pour l'accèlèrer encore plus. Voir Section 4.5.6.4 [SHOW VARIABLES], page 273.
- Si vous insèrez de nombreuses lignes depuis diffèrents clients, vous pouvez accèlèrer les insertions en utilisant la commande `INSERT DELAYED`. Voir Section 6.4.3 [`INSERT`], page 489.
- Notez qu'avec les tables `MyISAM`, vous pouvez insèrer des lignes en même temps que vous utilisez des commandes `SELECT`, du moment qu'il n'y a pas d'effacement de ligne dans la table.
- Lorsque vous chargez une table depuis un fichier texte, utilisez la commande `LOAD DATA INFILE`. Elle est gènèralement 20 fois plus rapide que l'èquivalent en commandes `INSERT`. Voir Section 6.4.9 [`LOAD DATA`], page 497.
- Il est possible, avec un peu de travail supplèmentaire, d'accèlèrer encore la vitesse des commandes `LOAD DATA INFILE`. Utilisez la procèdure standard :
  1. Crèez optionnellement une table avec `CREATE TABLE`. Par exemple, en utilisant `mysql` ou Perl-DBI.
  2. Exècutez une commande `FLUSH TABLES` ou la commande en ligne shell `mysqladmin flush-tables`.
  3. Utilisez `myisamchk --keys-used=0 -rq /path/to/db/tbl_name`. Cela va supprimer l'utilisation des index dans la table.
  4. Insèrez vos donnèes dans la table, avec `LOAD DATA INFILE`. Les index ne seront pas modifiès, et donc, trés rapides.
  5. Si vous allez uniquement lire la table dans le futur, utilisez `myisampack` pour la rèduire de taille. Voir Section 7.1.2.3 [Compressed format], page 537.
  6. Re-crèez les index avec `myisamchk -r -q /path/to/db/tbl_name`. Cette commande va crèer l'arbre d'index en mèmoire, avant de l'ècrire sur le disque, ce qui est bien plus rapide, car il n'y a que peu d'accés disques. L'arbre final sera aussi parfaitement èquilibrès.
  7. Exècutez une commande `FLUSH TABLES` ou utilisez la commande en ligne shell `mysqladmin flush-tables`.

  Notez que la ocmmande `LOAD DATA INFILE` fait aussi les optimisations ci-dessus, si vous faites les insertions dans une table vide. La diffèrence principale avec la procèdure ci-

dessus est que vous pouvez laisser `myisamchk` allouer plus de mèmoire temporaire pour la crèation d'index, que vous ne pourriez le faire pour chaque recrèation.

Depuis MySQL 4.0 vous pouvez aussi utiliser `ALTER TABLE tbl_name DISABLE KEYS` au lieu de `myisamchk --keys-used=0 -rq /path/to/db/tbl_name` et `ALTER TABLE tbl_name ENABLE KEYS` au lieu de `myisamchk -r -q /path/to/db/tbl_name`. De cette façon, vous pouvez aussi èviter l'ètape `FLUSH TABLES`.

- Vous pouvez accèlèrer les insertions qui sont faites avec plusieurs requêtes en verrouillant vos tables :

```
mysql> LOCK TABLES a WRITE;
mysql> INSERT INTO a VALUES (1,23),(2,34),(4,33);
mysql> INSERT INTO a VALUES (8,26),(6,29);
mysql> UNLOCK TABLES;
```

  La principale diffèrence de vitesse est que l'index de buffer est ècrit sur le disque une fois, aprés toutes les insertions `INSERT` terminèes. Normalement, il aurait du y avoir de nombreuses ècritures, une pour chaque commande `INSERT`. Le verrouillage n'est pas nècessaire si vous pouvez insèrer toutes les lignes d'une seule commande.

  Pour les tables transactionnelles, vous devriez utiliser `BEGIN/COMMIT` au lieu de `LOCK TABLES` pour accèlèrer les opèrations.

  Le verrouillage va aussi rèduire le nombre total de tests de connexions, mais le temps d'attente maximum de certains threads va augmenter (car il va y avoir la queue pour les verrous). Par exemple :

```
thread 1 fait 1000 insertions
thread 2, 3, et 4 font 1 insertion
thread 5 fait 1000 insertions
```

  Si vous ne voulez pas utiliser le verrouillage, les threads 2, 3 et 4 auront fini avant les 1 et 5. Si vous utilisez le verrouillage, 2, 3 et 4 me finiront probablement pas avant 1 ou 5, mais la durèe globale de l'opèration sera 40% plus courte.

  Comme les commandes `INSERT`, `UPDATE` et `DELETE` sont trés rapides avec MySQL, vous obtiendrez de meilleures performances gènèrales en ajoutant des verrous autour de toutes vos opèrations de 5 insertions o modifications simultanèes. Si vous faîtes de trés nombreux insertions dans une ligne, vous pouvez utiliser `LOCK TABLES` suivi de `UNLOCK TABLES` une fois de temps en temps (par exemple, toutes les 1000) pour permettre aux autres threads d'accèder á la table. Cela vous donnera quand même une bonne accèlèration.

  Bien sur, `LOAD DATA INFILE` reste bien plus rapide pour charger les donnèes.

Pour accèlèrer `LOAD DATA INFILE` et `INSERT`, agrandissez le buffer de clè. Voir Section 5.5.2 [Server parameters], page 390.

5.2.10 Vitesses des commandes `UPDATE`

Les requêtes de modification sont optimisèes comme les requêtes de `SELECT` avec le coût supplèmentaire de l'ècriture. La vitesse d'ècriture dèpend de la taille des donnèes qui sont modifièes, et du nombre d'index que cela va impacter. Les index ne sont pas modifiès tant que la ligne n'est pas ècrite. Les index qui ne sont pas modifiès ne seront pas rèècrit.

De plus, une autre mèthode pour obtenir des accèlèrations avec les modifications est de retarder les modifications, et d'en faire plusieurs d'un coup. Faire plusieurs modifications d'un coup est bien plus rapide que d'en faire une á chaque fois.

Notez que, avec le format de ligne dynamique, la modification d'une ligne peut dèboucher sur la fragmentation de la ligne. Si vous le faite souvent, il est trés important d'appliquer `OPTIMIZE TABLE` sur ces tables, pour les optimiser. Voir Section 4.5.1 [`OPTIMIZE TABLE`], page 264.

5.2.11 Rapiditè des requêtes `DELETE`

Si vous voulez effacer toutes les lignes d'une table, vous devez utiliser `TRUNCATE TABLE nom_de_table`. Voir Section 6.4.7 [TRUNCATE], page 496.

Le temps de suppression d'une ligne est exactement proportionnel au nombre d'index. Pour effacer les enregistrements plus rapidement, vous pouvez augmenter la taille du cache d'index. Voir Section 5.5.2 [Server parameters], page 390.

5.2.12 Autres conseils d'optimisation

Quelques conseils en vrac pour accèlèrer le serveur :

- Utilisez les connexions persistantes á la base, pour èviter les coûts rècurents de connexion. Si vous ne pouvez pas utiliser de connexions persistantes, et que vous faites de nombreuses connexions á la base, essayez de modifier la valeur de la variable `thread_cache_size`. Voir Section 5.5.2 [Server parameters], page 390.
- Vèrifiez toujours que vos requêtes utilisent vraiment les index que vous avez crèè dans les tables. Avec MySQL, vous pouvez utiliser la commande `EXPLAIN`. Voir Section 5.2.1 [Explain], page 363.
- Essayez d'èviter les requêtes `SELECT` complexes sur les tables `MyISAM` qui sont souvent modifièes. Cela èvitera des problémes de verrouillage.
- Les nouvelles tables `MyISAM` peuvent insèrer des lignes sans en effacer d'autre, tout en lisant dans cette table. Si c'est important pour vous, vous pouvez considèrer d'autres mèthodes oú vous n'avez pas á effacer de lignes, ou bien utilisez `OPTIMIZE TABLE` aprés avoir effacè beaucoup de lignes.
- Utilisez `ALTER TABLE ... ORDER BY expr1,expr2...` si vous lisez les colonnes dans l'ordre `expr1,expr2...`. Avec cette option, aprés de grosses modifications dans la table, vous pourriez obtenir de meilleures performances.
- Dans certains cas, cela vaut la peine d'ajouter une colonne qui est une combinaison ('hashed') des informations des autres colonnes. Si cette colonne est courte, et plutôt exemptes de doublons, elle peut se rèvèler plus rapide qu'un gros index sur plusieurs colonnes. Avec MySQL, il est trés facile d'utiliser une telle colonne : `SELECT * FROM table_name WHERE hash=MD5(CONCAT(col1,col2)) AND col_1='constant' AND col_2='constant'`
- Pour les tables qui sont souvent modifièes, vous devriez essayer d'èviter les colonnes `VARCHAR` et `BLOB`. Vous obtiendrez des lignes á format dynamique si vous utilisez ne serait-ce qu'une seule colonne `VARCHAR` ou `BLOB`. Voir Chapitre 7 [Table types], page 531.

- Normalement, cela ne sert á rien de sèparer une table en diffèrentes tables plus petites, juste parce que vos lignes deviennent 'grosses'. Pour accèder á une ligne, le plus long est le temps d'accés au premier octets de la ligne. Aprés cela, les disques modernes vont lire trés rapidement la ligne, et suffisament pour la plus par des applications. Le seul cas oú cela peut être important est si vous êtes capables de dègager une table á format de ligne fixe (voir ci-dessus), ou si vous avez besoin de scanner règuliérement la table, mais que vous n'avez pas besoin de toutes les colonnes. Voir Chapitre 7 [Table types], page 531.
- Si vous avez besoin de calculer souvent des expressions en fonction des informations placèes dans de nombreuses lignes (comme compter des lignes), il est probablement plus efficace d'introduire une nouvelle table qui va mettre á jour ce compteur en temps rèel. Une modification du type type `UPDATE table set count=count+1 where index_column=constant` est trés rapide!

  C'est trés important lorque vous utilisez les types de tables MyISAM et ISAM, qui ne dispose que d'un verrouillage de table (plusieurs lecteurs, un seul qui ècrit). Cela va aussi amèliorer les performances avec la plus par des bases, car le gestionnaire de verrouillage de ligne aura moins de tâches á faire.
- Si vous devez rassembler des statistiques issues de grosses tables de log, utiliser les tables de sommaires plutôt que la table compléte. Entretenir un sommaire est bien plus rapide que de regènèrer des tables á partir des logs á chaque modification (suivant la criticitè de vos informations), plutôt que de modifier l'application qui fonctionne.
- Si possible, essayez de marquer les rapports comme 'direct' ou 'statistique', oú les donnèes nècessaires pour les rapports statistiques ne sont gènèrèes qu'á partir de tables de sommaires, calculèes depuis les donnèes rèelles.
- Utilisez les valeurs par dèfaut des colonnes. N'insèrez des valeurs explicitement que lorsque la valeur différe de la valeur par dèfaut. Cela rèduit le temps d'analyse de MySQL, et amèliore les insertions.
- Dans certains cas, il est pratique de compacter et stocker les donnèes dans un blob. Dans ce cas, vous devez ajouter du code supplèmentaire pour compacter et dècompacter les donnèes dans le blob, mais cela pourra vous faire èconomiser de nombreux accés. C'est pratique lorsque vous avez des donnèes qui ne peuvent s'adapter facilement á une structure de base de donnèes.
- Normalement, vous devriez essayer de garder vos donnèes non redondantes (ce qui s'appelle la troisiéme forme normale dans les thèories de bases de donnèes), mais ne vous empêchez pas de duplique des donnèes ou de crèer des tables de sommaire, pour gagner de la vitesse.
- Les procèdures stockèes ou UDF (fonctions utilisateur) peuvent être une bonne façon de gagner en performance. Dans ce cas, vous devriez avoir une mèthode pour appliquer les mêmes fonctions d'une autre maniére, si votre base ne supporte les Les procèdures stockèes.
- Vous pouvez aussi gagner de la vitesse en utilisant des caches de requêtes dans vos applications, et en essayant de rassembler les nombreuses insertions ou modifications. Si votre base de donnèes supporte le verrouillage de table (comme MySQL et Oracle), cela vous aidera á vous assurer que le cache d'index est vidè aprés chaque modifications.

- Utilisez `INSERT /*! DELAYED */` lorsque vous n'avez pas besoin d'être assurè que vos donnèes sont ècrites. Cela accèlére les insertions, car de nombreuses lignes seront ècrites en une seule fois.
- Utilisez `INSERT /*! LOW_PRIORITY */` lorsque vous voulez que vos sèlections soient prioritaires.
- Utilisez `SELECT /*! HIGH_PRIORITY */` pour rendre les sèlections prioritaires. C'est á dire, les sèlections seront dèsormais faites même si un autre programme attend pour ècrire.
- Utilise la commande `INSERT` multiple pour insèrer plusieurs lignes en une seule commande SQL (plusieurs serveurs SQL le supporte).
- Utilisez `LOAD DATA INFILE` pour charger de grande quantitè de donnèes dans une table. C'est gènèralement plus rapide que des insertions, et sera même encore plus rapide une fois que `myisamchk` sera intègrè dans `mysqld`.
- Utilisez les colonnes `AUTO_INCREMENT` pour avoir des valeurs uniques.
- Utilisez `OPTIMIZE TABLE` une fois de temps en temps, pour èviter la fragmentation lors de l'utilisation de tables avec un format de ligne dynamique. Voir Section 4.5.1 [`OPTIMIZE TABLE`], page 264.
- Utilisez la tables de type `HEAP` pour accèlèrer les traitements au maximum. Voir Chapitre 7 [Table types], page 531.
- Avec un serveur web normal, les images doivent être stockèes dans des fichiers. C'est á dire, ne stockez qu'une rèfèrence au fichier d'image dans la base. La raison principale á cela est qu'un serveur web est bien meilleur pour mettre en cache des fichiers que le contenu d'une base de donnèes. Il est donc plus rapide si vous utilisez des fichiers.
- Utilisez des tables en mèmoire pour les donnèes non critiques, qui ont besoin d'être lues souvent (comme des informations sur la derniére banniére affichèe pour les utilisateurs sans cookies).
- Les colonnes contenant des informations identiques dans diffèrentes tables doivent être dèclarèes identiquement lors de la crèation des tables, et porter des noms identiques. Avant la version 3.23, vous pouviez ralentir les jointures.

  Essayez de garder des noms simples (utilisez `nom` au lieu de `nom_du_client` dans la table de clients). Pour rendre vos noms de colonnes portables vers les autres serveurs SQL, vous devriez essayer de les garder plus petits que 18 caractéres.
- Si vous avez vraiment besoin de trés haute vitesse, vous devriez considèrer les interfaces de bas niveau pour le stockage des donnèes que les diffèrents serveurs SQL supportent. Par exemple, en accèdant directement aux tables MySQL `MyISAM`, vous pourriez obtenir un gain de vitesse de l'ordre de 2 á 5 fois, en comparaison avec l'interface SQL. Pour cela, les donnèes doivent être sur le même serveur que l'application, et gènèralement, elles ne doivent être manipulèes que par un seul programme á la fois (car le verrouillage externe de fichiers est trés lent). Vous pouvez èliminer ces problémes en crèan des commandes `MyISAM` de bas niveau dans le serveur MySQL (cela peut se faire facilement pour amèliorer les performances). Soyez trés prudent dans la conception de votre interface, mais il est trés facile de supporter ce type d'optimisation.
- Dans de nombreux cas, il est plus rapide d'accèder aux donnèes depuis une base (en utilisant une connexion ouverte) que d'accèder á un fichier texte, car la base de donnèes

est plus compacte que le fichier texte (si vous utilisez des donnèes numèriques), et cela entraine moins d'accés disques. Vous allez aussi èconomiser du code, car vous n'aurez pas á analyser le fichier texte pour repèrer les limites de lignes.

- Vous pouvez aussi utiliser la rèplication pour accèlèrer le serveur. Voir Section 4.10 [Replication], page 332.
- Dèclarer une table avec `DELAY_KEY_WRITE=1` va accèlèrer la mise á jour des index, car ils ne seront pas ècrit sur le disque jusqu'á ce que le fichier de donnèes soit refermè. L'inconvènient est que vous devez exècuter l'utilitaire `myisamchk` sur ces tables avant de lancer `mysqld` pour vous assurer que les index sont bien á jour, au cas oú le processus aurait ètè interrompu avant d'enregistrer les donnèes. Comme les informations d'index sont toujours regènèrables, vous ne perdrez pas de donnèes avec `DELAY_KEY_WRITE`.

5.3 Verrouillage de tables

5.3.1 Comment MySQL verrouille les tables

Vous pouvez trouver une discussion sur les diffèrentes mèthodes de verrouillage dans l'annexe. Voir Section E.4 [Locking methods], page 823.

Toutes les mèthodes de verrouillage de MySQL sont exemptes de blocage, sauf pour les tables `InnoDB` et `BDB`. Ceci fonctionne en demandant tous les verrous d'un seul coup, au dèbut de la requête, et en verrouillant les tables toujours dans le même ordre.

Les tables `InnoDB` obtiennent automatiquement leur verrou de ligne et les tables `BDB` leur verrou de page, durant le traitement de la requête SQL, et non pas au dèmarrage de la transaction.

La mèthode de verrouillage des tables de MySQL en ècriture (`WRITE`) fonctionne comme ceci :

- Si il n'y a pas de verrou sur la table, pose un verrou en ècriture dessus.
- Sinon, soumet une requête de verrouillage dans la queue de verrous d'ècriture.

La mèthode de verrouillage des tables de MySQL en lecture (`READ`) fonctionne comme ceci :

- Si il n'y a pas de verrou sur la table, pose un verrou en ècriture dessus.
- Sinon, soumet une requête de verrouillage dans la queue de verrou de lecture.

Lorsqu'un verrou est libèrè, le verrou est donnè aux threads de la queue de verrou en ècriture, puis á ceux de la queue de verrou en lecture.

Cela signifie que si vous avez de nombreuses modifications dans une table, la commande `SELECT` va attendre qu'il n'y ait plus d'ècriture avant de lire.

Pour contourner ce probléme dans les cas oú vous voulez faire de nombreuses `INSERT` et `SELECT` sur la même table, vous pouvez insèrer les lignes dans une table temporaire, et ne modifier la table rèelle que de temps en temps, á partir de la table temporaire.

Ceci peut être fait comme ceci :

```
mysql> LOCK TABLES real_table WRITE, insert_table WRITE;
mysql> INSERT INTO real_table SELECT * FROM insert_table;
mysql> TRUNCATE TABLE insert_table;
mysql> UNLOCK TABLES;
```

Vous pouvez aussi utiliser les options `LOW_PRIORITY` de `INSERT`, `UPDATE` ou `DELETE` ou `HIGH_PRIORITY` avec `SELECT` si vous voulez donner d'autres prioritèes aux actions. Vous pouvez aussi dèmarrer `mysqld` avec l'option `--low-priority-updates` pour obtenir ce type de comportement.

Utiliser l'option `SQL_BUFFER_RESULT` peut aussi rèduire la durèe des verrous de tables. Voir Section 6.4.1 [SELECT], page 481.

Vous pouvez aussi changer le code de verrouillage dans le fichier '`mysys/thr_lock.c`', pour utiliser une queue simple. Dans ce cas, les verrous en ècriture et en lecture auront la même prioritè, ce qui peut aider certaines applications.

5.3.2 Probléme de verrouillage de tables

Le systéme de verrou de MySQL est exempt de blocage.

MySQL utilise le verrouillage de table (au lieu du verrouillage de ligne ou de colonne) sur tous les types de tables, sauf `InnoDB` et `BDB`, pour obtenir un systéme de verrou á trés haute vitesse. Pour les grandes tables, le verrouillage de table est bien plus rapide que le verrouillage de page, mais il y a aussi des inconvènients.

Pour les tables `InnoDB` et `BDB`, MySQL n'utilise le verrouillage de table que vous le demandez explicitement avec `LOCK TABLES`. Pour ces tables, nous vous recommandons de ne jamais utiliser la commande `LOCK TABLES`, car `InnoDB` utilise un verrouillage de ligne automatique, et `BDB` utilise un verrouillage de pages, pour assurer l'isolation des transactions.

Depuis MySQL version 3.23.7, vous pouvez insèrer des lignes dans les tables `MyISAM`, en même temps que d'autres threads y lisent. Notez que, actuellement, cela ne fonctionne que si il n'y a pas de trous aprés les lignes effacèes dans la table, au moment de l'insertion. Lorsque tous les trous ont ètè bouchès avec de nouvelles donnèes, les insertions simultanèes seront automatiquement rèactivèes.

Le verrouillage de table permet á de nombreux threads de lire dans la même table, mais si un thread dèsire ècrire dans la table, il doit obtenir un verrou en ècriture pour avoir un accés exclusif. Durant la modification, les autres threads qui voudront lire dans cette table, devront attendre.

Comme les modifications de tables sont considèrèes comme plus importantes que les lectures avec `SELECT`, toutes les commandes qui modifient la table ont prioritès sur les lectures. Cela devrait vous assurer que les modifications ne sont pas retenues trop longtemps, á cause de nombreuses lectures sur une même table. Vous pouvez toutefois modifier cela avec l'option `LOW_PRIORITY` des commandes de modification, et l'option `HIGH_PRIORITY` de `SELECT`).

Depuis MySQL version 3.23.7, vous pouvez utiliser la variable `max_write_lock_count` pour forcer MySQL á laisser temporairement la place á toutes les commandes `SELECT`, aprés un certain nombre de modifications dans la table.

Le verrouillage de table est une mauvaise technique dans les situations suivantes :

- Un client exècute une commande `SELECT` qui prend trés longtemps.

- Un autre client exècute une commande `UPDATE` sur la table. Ce client va devoir attendre que la commande `SELECT` soit finie.
- Un autre client exècute une autre commande `SELECT` sur la même table. Comme `UPDATE` a la prioritè sur `SELECT`, cette commande `SELECT` va attendre que `UPDATE` soit finit. Il va donc attendre que le premier `SELECT` soit fini.
- Un thread attend des erreurs comme `full disk`, auquel cas tous les threads qui veulent accèder á la table seront en attente de plus d'espace sur le disque.

Des solutions aux problémes sont :

- Essayez d'accèlèrer au maximum les commandes `SELECT`. Vous pourriez passer par une table de sommaire pour cela.
- Dèmarrez `mysqld` avec l'option `--low-priority-updates`. Cela va donner aux commandes de modification une prioritè plus faible que `SELECT`. Dans ce cas, c'est la commande `SELECT` du prècèdent scènario qui s'exècutera avant la commande `INSERT`.
- Vous pouvez donner á une commande spècifique `INSERT`, `UPDATE` ou `DELETE`, une prioritè plus basse avec l'attribut `LOW_PRIORITY`.
- Dèmarrez `mysqld` avec une valeur faible pour `max_write_lock_count` afin de donner plus souvent la chance aux verrous `READ` la possibilitè de lire des donnèes, entre deux verrous `WRITE`.
- Vous pouvez spècifier que toutes les modifications d'un thread spècifique doivent être faites avec un prioritè basse, en utilisant la commande SQL : `SET LOW_PRIORITY_UPDATES=1`. Voir Section 5.5.6 [SET], page 396.
- Vous pouvez spècifier qu'une requête particuliére `SELECT` est trés importante, en utilisant l'attribut `HIGH_PRIORITY`. Voir Section 6.4.1 [SELECT], page 481.
- Si vous avez des problémes avec des `INSERT` combinès avec des `SELECT`, utilisez les tables `MyISAM` car elle supportent les commandes `SELECT`s et `INSERT` simultanèes.
- Si vous voulez mèlanger les commandes `INSERT` et `SELECT`, utilisez l'attribut `DELAYED` de la commande `INSERT` pour rèsoudre ce probléme. Voir Section 6.4.3 [INSERT], page 489.
- Si vous avez des problémes avec `SELECT` et `DELETE`, l'option `LIMIT` de `DELETE` peut aider. Voir Section 6.4.6 [DELETE], page 494.

5.4 Optimisation de la structure de la base de donnèes

5.4.1 Conception

MySQL conserve les donnèes et les index dans deux fichiers sèparès. De nombreux (et en fait presque toutes) les autres bases mèlangent les donnèes et les index dans le même fichier. Nous pensons que le choix de MySQL est bien meilleur pour un grand nombre de systémes modernes.

Une autre mèthode de stockage des donnèes est de conserver les informations de chaque colonne dans une zone sèparèe (par exemple SDBM et Focus). Cela va rèduire les performances qui accédent á plus d'une colonne. Comme cela dègènére vite lorsque plus d'une colonne est utilisèe, nous pensons que ce modéle n'est pas bon pour une base de donnèes gènèraliste.

Les cas les plus courants sont que les index et les donnèes sont stockèes ensemble (comme Oracle/Sybase et al). Dans ce cas, vous aurez aussi les informations de lignes dans la page finale de l'index. L'intèrêt d'une telle organisation est que, dans de nombreuses situations, dèpendamment du cache d'index, vous èconomisez des lectures disques. Les problémes de cette organisation sont :

- Le scan des tables est bien plus lent, car vous devez lire les index pour obtenir les donnèes.
- Vous ne pouvez pas utiliser uniquement l'index pour lire des donnèes pour une requête.
- Vous utilisez beaucoup d'espace, et vous devez dupliquer des index de noeuds (car vous ne pouvez pas simplement stocker des lignes dans les noeuds).
- Les suppressions vont perturber la table (comme les index ne sont pas modifiès lors de l'effacement).
- Il est plus difficile de ne mettre en cache que les donnèes.

5.4.2 Rendre vos tables aussi compactes que possible

Une des optimisations simple est de rèduire au maximum la taille de vos donnèes et de vos index sur le disque et en mèmoire. Cela peut donner des accèlèrations impressionnantes, car les lectures sur le disque sont plus rapides, et moins de mèmoire centrale sera utilisèe. L'indexation de colonnes de petites taille prend aussi moins de ressources.

MySQL supporte un grand nombre de type de tables et de format de ligne. Choisir ces types peut vous conduire á des amèliorations de performances. Voir Chapitre 7 [Table types], page 531.

Vous pouvez obtenir des gains de performances sur les tables et minimiser l'espace disque en utilisant les techniques ci-dessous :

- Utilisez les types les plus efficaces et les plus petits possibles. MySQL a diffèrents types spècialisès qui èpargnent de l'espace disque et de la mèmoire.
- Utilisez les types d'entiers les plus petits possible pour rèduire les tables. Par exemple, `MEDIUMINT` est souvent prèfèrable á `INT`.
- Dèclarez les colonnes pour qu'elle soient `NOT NULL` si possible. Cela accèlére les traitements, et vous fait gagner un bit par colonne. Notez que si vous avez vraiment besoin d'une valeur `NULL` dans votre application, il est recommandè de l'utiliser. Evitez simplement de l'utiliser par dèfaut sur toutes les colonnes.
- Si vous n'avez pas de colonne de taille variable (`VARCHAR`, `TEXT` ou `BLOB`), un format de ligne á taille fixe est utilisè. C'est plus rapide, mais cela prend plus d'espace sur le disque. Voir Section 7.1.2 [`MyISAM` table formats], page 535.
- La clè primaire doit être aussi courte que possible. Cela rend l'identification des lignes plus efficace.
- Pour chaque table, vous devez dècider quel mèthode de stockage et d'indexation vous allez utiliser. Voir Chapitre 7 [Table types], page 531.
- Ne crèez que des index dont vous avez besoin. Les index sont bons pour accèlèrer les lectures, mais sont plus lents lorsque vous ècrivez des donnèes. Si vous accèder essentiellement á votre table en lecture avec des combinaisons de colonnes, faites un

index avec ces colonnes. Le premier index doit être la colonne la plus utilisèe. Si vous utilisez **constamment** de nombreuses colonnes, vous devriez utiliser la colonne avec le plus de doublons en premier, pour obtenir une meilleure compression.

- Si il est probable qu'une colonne a un prèfixe unique avec les premiers caractéres, il est mieux de n'indexer que ce prèfixe. MySQL supporte les index sur une partie de colonne. Les index les plus courts sont les plus efficaces car ils prennent moins d'espace disque, et aussi, car ils absorbent plus de requêtes grâce au cache en mèmoire. Voir Section 5.5.2 [Server parameters], page 390.
- Dans certaines circonstances, il peut être intèressant de sèparer en deux une table qui est scannèe trés souvent. C'est particuliérement vrai pour les formats de tables dynamiques, et si possible, utilisez un format de table statique pour les colonnes les plus pertinentes.

5.4.3 Comment MySQL utilise les index

Les index sont utilisès pour trouver des lignes de rèsultat avec une valeur spècifique, trés rapidement. Sans index, MySQL doit lire successivement toutes les lignes, et á chaque fois, faire les comparaisons nècessaires pour extraire un rèsultat pertinent. Plus la table est grosse, plus c'est coûteux. Si la table dispose d'un index pour les colonnes utilisèes, MySQL peut alors trouver rapidement les positions des lignes dans le fichier de donnèes, sans avoir á fouiller toute la table. Si une table á 1000 lignes, l'opèration sera alors 100 fois plus rapide qu'une lecture sèquentielle. Notez que si vous devez lire la presque totalitè des 1000 lignes, la lecture sèquentielle se rèvèlera alors plus rapide, malgrè tout.

Tous les index de MySQL (`PRIMARY`, `UNIQUE` et `INDEX`) sont stockès sous la forme de B-trees. Les chaînes sont automatiquement prèfixèe et leurs espaces terminaux sont supprimès. Voir Section 6.5.7 [`CREATE INDEX`], page 517.

Les index sont utilisès pour :

- Trouver rapidement des lignes qui satisfont une clause `WHERE`.
- Lire des lignes dans d'autres tables lors des jointures.
- Trouver les valeurs `MAX()` et `MIN()` pour une colonne indexèe. C'est une opèration qui est optimisèe par le prèprocesseur, qui vèrifie si vous utilisez la constante `WHERE` key_part_# = sur toute les parties de clès infèrieures á `<` N. Dans ce cas, MySQL va faire une simple recherche de clè et remplacer l'expression par une constante. Si toutes les expressions sont remplacèes par des constantes, la requête va alors être rapidement calculèe :

```
SELECT MIN(key_part2),MAX(key_part2) FROM table_name where key_part1=10
```

- Trier ou grouper des lignes dans une table, si le tri ou le regroupement est fait avec un prèfixe á gauche utilisable (par exemple, `ORDER BY key_part_1,key_part_2`). La clè est lue en ordre inverse, si toutes les parties de clès sont suivies du mot clè `DESC`. Voir Section 5.2.7 [ORDER BY optimisation], page 372.
- Dans certains cas, la requête peut être optimisèe pour lire des valeurs sans consulter le fichier de donnèes. Si cette possibilitè est utilisèe avec des colonnes qui sont toutes numèriques, et forme le prèfixe de gauche d'une clè, les valeurs peuvent être lues depuis l'index, á grande vitesse :

```
SELECT key_part3 FROM table_name WHERE key_part1=1
```

Supposez que vous utilisiez la commande SELECT suivante :

```
mysql> SELECT * FROM tbl_name WHERE col1=val1 AND col2=val2;
```

Si un index multi-colonne existe sur les colonnes col1 et col2, les lignes appropriées seront directement lues. Si des index sèparès sur les colonnes col1 et col2 existent, l'optimiseur va essayer de trouver l'index le plus restrictif des deux, en dècidant quel index dèbouche sur le moins de lignes possibles.

Si une table a un index multi-colonne, tout prèfixe d'index peut être utilisè par l'optimiseur pour trouver des lignes. Par exemple, si vous avez un index á trois colonnes (col1,col2,col3), vous pouvez faire des recherches accèlèrèes sur les combinaisons de colonnes (col1), (col1,col2) et (col1,col2,col3).

MySQL ne peut utiliser d'index partiel sir les colonnes ne forment pas un prèfix d'index. Supposez que vous avez la commande SELECT suivante :

```
mysql> SELECT * FROM tbl_name WHERE col1=val1;
mysql> SELECT * FROM tbl_name WHERE col2=val2;
mysql> SELECT * FROM tbl_name WHERE col2=val2 AND col3=val3;
```

Si un index existe sur les colonnes (col1,col2,col3), seule la premiére requête pourra utiliser l'index ci-dessus. Les deux autres requêtes utilisent des colonnes indexèes, mais les colonnes (col2) et (col2,col3) ne font pas partie du prèfixe des colonnes (col1,col2,col3).

MySQL utilise aussi les index lors des comparaisons avec l'opèrateur LIKE si l'argument de LIKE est une chaîne constante qui ne commence pas par un caractére joker. Par exemple, les requêtes SELECT suivantes utilisent des index :

```
mysql> SELECT * FROM tbl_name WHERE key_col LIKE "Patrick%";
mysql> SELECT * FROM tbl_name WHERE key_col LIKE "Pat%_ck%";
```

Dans le premier exemple, seules les lignes avec "Patrick" <= key_col < "Patricl" sont considèrèes. Dans le second exemple, "Pat" <= key_col < "Pau" sont considèrèes.

Les commandes SELECT suivantes n'utilisent pas d'index :

```
mysql> SELECT * FROM tbl_name WHERE key_col LIKE "%Patrick%";
mysql> SELECT * FROM tbl_name WHERE key_col LIKE other_col;
```

Dans la premiére requête, la valeur associèe á LIKE commence avec un caractére joker. Dans le second exemple, la valeur associèe á LIKE n'est pas une valeur constante.

MySQL 4.0 fait une autre optimisation avec l'opèrateur LIKE. Si vous utilisez ... LIKE "%string%" et que string est plus grand que 3 caractéres, MySQL va utiliser l'algorithme Turbo Boyer-Moore qui prend une valeur initiale pour rèsoudre le masque, et l'exploite pour accèlèrer la recherche.

Les recherches qui utilisent la fonction column_name IS NULL vont utiliser les index si column_name sont des index.

MySQL normalement utilise l'index qui gènére le moins de lignes possible. Un index est utilisè avec les colonnes que vous spècifiez, et les opèrateurs suivants : =, >, >=, <, <=, BETWEEN et l'opèrateur LIKE sans prèfixe joker, c'est á dire de la forme 'quelquechose%'.

Aucun n'index qui ne s'applique pas á tous les niveaux de AND dans une requête WHERE, ne sera pas utilisè pour optimiser la requête. En d'autres termes, pour être capable d'utiliser

un index pour optimiser une requête, un prèfixe de l'index doit être utilisè dans toutes les parties de la formule logique contenant `AND`.

Les clauses `WHERE` suivantes utilisent des index :

```
... WHERE index_part1=1 AND index_part2=2 AND other_column=3
... WHERE index=1 OR A=10 AND index=2      /* index = 1 OR index = 2 */
... WHERE index_part1='hello' AND index_part_3=5
          /* optimisè par "index_part1='hello'" */
... WHERE index1=1 and index2=2 or index1=3 and index3=3;
          /* peut utiliser un index sur index1 mais pas sur index2 ou index 3 */
```

Ces clauses `WHERE` n'utilisent **PAS** d'index :

```
... WHERE index_part2=1 AND index_part3=2  /* index_part_1 n'est pas utilisè */
... WHERE index=1 OR A=10                  /* Index n'est pas utilisè sur les deux
... WHERE index_part1=1 OR index_part2=10  /* Aucun index ne s'applique á toutes le
```

Notez que dans certains cas, MySQL ne va pas utiliser un index, même si il en reste de disponible. Voici certains de ces cas :

- Si l'utilisation de l'index requiert que MySQL accéde á plus de 30% des lignes de la table (dans ce cas, un scan de table est probablement plus rapide, et demandera moins d'accés disques). Notez que si une telle requête utilise la clause `LIMIT` pour ne lire qu'une partie des lignes, MySQL utilisera tout de même l'index, car il va trouver plus rapidement les quelques lignes de rèsultat.
- Si un intervalle d'index contient des valeurs `NULL` et que vous utilisez la clause `ORDER BY ... DESC`.

5.4.4 Index de colonnes

Tous les types de colonnes ed MySQL peuvent être indexès. L'utilisation des index sur les colonnes pertinentes est la meilleur façon d'amèliorer les performances de opèrations de `SELECT`.

Le nombre maximum de clefs et la longueur maximale des index sont dèfinis pour chaque type de table. Voir Chapitre 7 [Table types], page 531. Vous pouvez avec tous les gestionnaires de tables avoir au moins 16 clefs et une taille totale d'index d'au moins 256 octets.

Pour les colonnes `CHAR` et `VARCHAR` , il est possible d'indexer un prèfixe de la colonne. C'est plus rapide et plus èconome en espace disque que l'indexation de la colonne entiére. La syntaxe pour indexer le dèbut d'une colonne au moment de la crèation de la table ressemble á cela:

```
KEY nom_de_l_index (nom_de_la_colonne(longueur))
```

L'exemple suivant crèe un index sur les 10 premiers caractéres de la colonne `nom` :

```
mysql> CREATE TABLE test (
    ->        nom CHAR(200) NOT NULL,
    ->        KEY index_du_nom (nom(10)));
```

Pour les colonnes `BLOB` et `TEXT`, vous devez indexer le dèbut de la colonne. Vous ne pouvez pas indexer une colonne entiére.

Dans les versions 3.23.23 et supèrieures de MySQL, vous pouvez aussi crèer les index spèciaux `FULLTEXT`. Ils sont utilisè pour les recherches sur les textes en entier. Seules les tables de type `MyISAM` supportent les index `FULLTEXT`. Il peuvent être crèès depuis des colonnes `CHAR`, `VARCHAR` et `TEXT`. L'indexation se fait toujours par rapport á la colonne en entier, et l'indexation partielle n'est pas supportèe. Voir Section 6.8 [Fulltext Search], page 522 pour plus d'informations.

5.4.5 Index sur plusieurs colonnes

MySQL peut crèer des index sur plusieurs colonnes. Un index peut comprendre jusqu'á 15 colonnes. (sur les colonnes de type `CHAR` ou `VARCHAR`, vous pouvez utiliser uniquement le dèbut de la colonne pour l'indexation.)

Un index sur plusieurs colonnes peut être compris comme un tableau triè contenant des valeurs crèèes par concatènation des valeurs des colonnes indexèes.

MySQL utilise les index sur plusieurs colonnes de telle sorte que les requêtes sont accèlèrèes quand on spècifie une quantitè connue de la premiére colonne de l'index dans un clause `WHERE`, même si on ne spècifie pas la valeur des autres colonnes.

On suppose qu'une table est crèèe avec les paramétres suivant:

```
mysql> CREATE TABLE test (
    ->       id INT NOT NULL,
    ->       nom CHAR(30) NOT NULL,
    ->       prenom CHAR(30) NOT NULL,
    ->       PRIMARY KEY (id),
    ->       INDEX nom_index (nom,prenom));
```

Alors l'index `nom_index` est un index de `nom` et de `prenom`. Cela sera utile pour les requêtes qui spècifient des valeurs dans une gamme donnèe de `nom`, ou pour á la fois `nom` et `prenom`. Ainsi l'index `nom_index` sera utilisè pour les requêtes suivantes:

```
mysql> SELECT * FROM test WHERE nom="Widenius";

mysql> SELECT * FROM test WHERE nom="Widenius"
    ->                     AND prenom="Michael";

mysql> SELECT * FROM test WHERE nom="Widenius"
    ->                     AND (prenom="Michael" OR prenom="Monty");

mysql> SELECT * FROM test WHERE nom="Widenius"
    ->                     AND prenom >="M" AND prenom < "N";
```

Cependant, l'index `nom_index` ne sera PAS utilisè pour les requêtes suivantes:

```
mysql> SELECT * FROM test WHERE prenom="Michael";

mysql> SELECT * FROM test WHERE nom="Widenius"
    ->                     OR prenom="Michael";
```

Pour plus d'informations sur la mèthode de MySQL pour utiliser les index dans le but d'amèliorer les performance des requêtes, regardez Section 5.4.3 [MySQL indexes], page 384.

5.4.6 Pourquoi tant de tables ouvertes ?

Quand vous utiliserez la commande `mysqladmin status`, vous verrez quelque chose de ce genre :

```
Uptime: 426 Running threads: 1 Questions: 11082 Reloads: 1 Open tables: 12
```

Cela vous laissera perplexe si vous n'avez que 6 tables.

MySQL est multi-threadè, il peut donc exècuter plusieurs requêtes sur la même table simultanèment. Pour minimiser les interfèrences entre deux threads ayant diffèrentes actions sur le même fichier, la table est ouverte indèpendamment par chacun des threads. Cela nècessite un peu de mèmoire, mais augmente les performances. Avec les tables au format `ISAM` et `MyISAM`, cela requiére aussi un fichier additionnel de description du fichier des donnèes. Avec ce type de tables, le fichier dècrivant l'index est partagè entre tous les threads.

Vous pourrez lire plus sur le sujet á la section suivante : Voir Section 5.4.7 [Table cache], page 388.

5.4.7 Quand MySQL ouvre et ferme les tables

`table_cache`, `max_connections` et `max_tmp_tables` affectent le nombre maximum de tables que le serveur garde ouvertes. Si vous augmentez l'une de ces valeurs, vous pourriez rencontrer une des limites de votre systéme d'exploitation. Cependant, vous pourrez augmenter ces limites sur de nombreux systémes d'exploitation. Consultez votre documentation systéme pour voir comment faire cela, car la mèthode pour modifier la limite est diffèrente pour chaque systéme.

`table_cache` est liè au `max_connections`. Par exemple, pour 200 connexions simultanèes, vous devriez avoir un cache de table d'environ `200 * n`, oú `n` est le nombre maximum de table dans une jointure. Vous devez aussi rèserver des pointeurs de fichiers supplèmentaires pour les tables temporaires et les fichiers.

Assurez vous que votre systéme d'exploitation peut gèrer le nombre de pointeurs de fichiers demandè par l'option `table_cache`. Si `table_cache` est trop grand, MySQL peut être á court de pointeurs, et refuser des connexions, èchouer á l'exècution de requêtes, ou être trés instable. Vous devez aussi prendre en compte que les tables `MyISAM` peuvent avoir besoin de deux pointeurs de fichiers pour chaque table diffèrente. Vous pouvez augmenter le nombre de pointeurs de fichiers disponibles pour MySQL avec l'option de dèmarrage `--open-files-limit=#`. Voir Section A.2.16 [Not enough file handles], page 694.

Le cache de tables ouvertes reste au niveau de `table_cache` entrèes (par dèfaut, 64; cela peut être modifiè avec l'option `-O table_cache=#` de `mysqld`). Notez que MySQL peut ouvrir temporairement plus de tables, pour être capable d'exècuter des requêtes.

Une table qui n'est pas utilisèe est refermèe, et supprimèe du cache de table, dans les circonstances suivantes :

- Lorsque le cache est plein, et qu'un thread essaie d'ouvrir une table qui n'est pas dans le cache.
- Lorsque le cache contient plus de `table_cache` lignes, et qu'aucun thread n'utilise cette table.

- Lorsque quelqu'un utilise la commande `mysqladmin refresh` ou `mysqladmin flush-tables`.
- Lorsque quelqu'un exècute la commande `FLUSH TABLES`.

Lorsque le cache de table se remplit, le serveur utilise la procèdure suivante pour identifier une entrèe du cache, pour la supprimer :

- Les tables qui n'est pas en cours d'utilisation est libèrèe, en utilisant la table qui a ètè accèdè depuis plus longtemps en premier.
- Si le cache est plein, et qu'aucune table ne peut être libèrèe, mais qu'une nouvelle table doit être ouverte, le cache est temporairement ètendu.
- Si le cache est dans un ètat d'extension, et qu'une table passe de l'ètat d'utilisation á non utilisation, la table est immèdiatement fermèe et libèrèe du cache.

Une table est ouverte pour chaque accés simultanè. Cela signifie que si vous avez deux threads qui accédent á la même table, ou accédent á la même table deux fois dans la requête (avec `AS`), la table devra être ouverte deux fois. La premiére ouverture d'une table prendre deux pointeurs de fichiers. Chaque utilisation supplèmentaire de la même table ne prendra qu'un pointeur supplèmentaire. Le pointeur de fichier supplèmentaire de la premiére table est celui du fichier d'index. Ce pointeur est partagè entre les threads.

Si vous ouvrez une table avec `HANDLER table_name OPEN`, un objet de table dèdiè sera allouè pour le thread. Cet objet de table n'est pas partagè avec les autres threads, et il ne sera pas fermè avant que le thread n'appelle `HANDLER table_name CLOSE`, ou que le thread ne meurt. Voir Section 6.4.2 [HANDLER], page 488. Lorsque cela arrive, la table est placèe dans le cache de table (si il n'est pas plein).

Vous pouvez vèrifier si votre cache de table n'est pas trop petit en vèrifiant la variable de `mysqld` appelèe `Opened_tables`. Si cette valeur est grande, même si vous n'avez pas trop abusè de la commande `FLUSH TABLES`, vous devrez augmenter la taille du cache. Voir Section 4.5.6.3 [SHOW STATUS], page 269.

5.4.8 Inconvènients de la crèation d'un grand nombre de tables dans la même base de donnèes

Si vous avez beaucoup de fichiers dans un dossier, les opèrations d'ouverture, fermeture, et crèation seront ralenties. Si vous exècutez une requête `SELECT` sur plusieurs tables, il y'aura une lègére perte lorsque le cache de tables sera plein, car pour chaque table ouverte, une autre doit être fermèe. Vous pouvez rèduire cette table en augmentant la taille du cache de tables.

5.5 Optimisation du serveur MySQL

5.5.1 Règlage du systéme, au moment de la compilation, et paramétres du dèmarrage

Nous dèmarrons par le niveau du systéme, car certaines dècisions á ce niveau doivent être prises trés tôt. Dans d'autres cas, un regard rapide á cette partie doit suffire, car ce n'est

pas tellement important pour les gros gains. Toute fois, il est toujours sympathique de sentir combien on peut gagner en changeant des choses á ce niveau.

Le choix du systéme d'exploitation est vraiment important! Pour utiliser au maximum les capacitès de machines multi-processeurs, il vaut mieux choisir Solaris (car les threads marchent vraiment trés bien) ou Linux (car le noyau 2.2 supporte trés bien le SMP). Mais les plate-formes Linux 32 bits limitent par dèfaut la taille des fichiers á 2 Go. Heureusement, cela sera bientot rèparè avec l'arrivèe des nouveaux systémes de fichier (XFS/Reiserfs). Si vous souhaitez dèsespèremment utiliser des fichiers de plus de 2 Go sur Linux-intel 32bits, vous devriez utiliser le patch de LFS pour le systéme de fichier ext2

Comme nous n'avons pas utilisè MySQL en production sur ènormèment de plate-formes, nous vous conseillons de tester votre plate-forme avant de la choisir dèfinitivement.

Autres astuces:

- Si vous avez suffisamment de RAM, vous pouvez supprimer toutes les partitions d'èchange (swap). Certains systémes d'exploitation utilisent parfois la partition d'èchange quand bien même il reste de la mèmoire libre.
- L'utilisation de l'option `--skip-external-locking` de MySQL empêche les verrous externes. Cela n'influencera pas les fonctionnalitès de MySQL tant que vous n'utilisez qu'un seul serveur. Il faut cependant penser á arreter le serveur (ou bien de verrouiller les parties pertinantes) avant d'utiliser `myisamchk`. Sur certains systémes, cette option est inutile car les verrous externes ne fonctionnent pas du tout.

  L'option `--skip-external-locking` est activèe par dèfaut quand on compile avec MIT-pthreads, car `flock()` n'est pas totalement supportè sur toutes les plate-formes par MIT-pthreads. Elle l'est ègalement sur Linux, cat le verrouillage des fichiers de Linux n'est pas encore sûr.

  Les seuls cas oú on ne peut pas utiliser `--skip-external-locking` sont si on utilise plusieurs *servers* (pas de clients) MySQL sur les mêmes donnèes, ou si on lance `myisamchk` sur une table sans vider son tampon et sans la verrouiller au prèalable.

  Il est toujours possible d'utiliser `LOCK TABLES/UNLOCK TABLES` même si vous utilisez `--skip-external-locking`.

5.5.2 Règlage des paramétres du serveur

Vous pouvez obtenir les tailles par dèfaut des tampons du serveur `mysqld` avec la commande:

```
shell> mysqld --help
```

Cette commande gènére une liste de toutes les options de `mysqld` et des variables configurables. Cette sortie comprend les valeurs par dèfaut et ressemble á cela:

```
Possible variables for option --set-variable (-O) are:
back_log                current value: 5
bdb_cache_size          current value: 1048540
binlog_cache_size       current value: 32768
connect_timeout         current value: 5
delayed_insert_timeout  current value: 300
delayed_insert_limit    current value: 100
delayed_queue_size      current value: 1000
```

```
flush_time                current value: 0
interactive_timeout       current value: 28800
join_buffer_size          current value: 131072
key_buffer_size           current value: 1048540
lower_case_table_names    current value: 0
long_query_time           current value: 10
max_allowed_packet        current value: 1048576
max_binlog_cache_size     current value: 4294967295
max_connections           current value: 100
max_connect_errors        current value: 10
max_delayed_threads       current value: 20
max_heap_table_size       current value: 16777216
max_join_size             current value: 4294967295
max_sort_length           current value: 1024
max_tmp_tables            current value: 32
max_write_lock_count      current value: 4294967295
myisam_sort_buffer_size   current value: 8388608
net_buffer_length         current value: 16384
net_retry_count           current value: 10
net_read_timeout          current value: 30
net_write_timeout         current value: 60
read_buffer_size          current value: 131072
record_rnd_buffer_size    current value: 131072
slow_launch_time          current value: 2
sort_buffer               current value: 2097116
table_cache               current value: 64
thread_concurrency        current value: 10
tmp_table_size            current value: 1048576
thread_stack              current value: 131072
wait_timeout              current value: 28800
```

Il est á noter que `--set-variable` n'est plus utile depuis MySQL 4.0, et on peut utiliser `--var=option` directement.

Si un serveur `mysqld` est en cours d'exècution, vous pouvez voir les valeurs que les variables utilisent rèellement en exècutant la commande :

```
shell> mysqladmin variables
```

Vous pouvez trouver une description compléte de toutes les variables dans la section `SHOW VARIABLES` de ce manuel. Voir Section 4.5.6.4 [SHOW VARIABLES], page 273.

Vous pouvez aussi voir des statistiques sur un serveur en cours d'exècution en utilisant la commande `SHOW STATUS`. Voir Section 4.5.6.3 [SHOW STATUS], page 269.

MySQL utilise des algorithmes trés extensibles, donc vous pouvez utiliser trés peu de mèmoire. Si malgrè tout vous fournissez plus de mèmoire á MySQL, vous obtiendrez ègalement de meilleures performances.

Les deux variables les plus importantes au moment du règlage d'un serveur MySQL sont `key_buffer_size` et `table_cache`. Vous devriez vous assurer que celles sont sont bien paramètrèes avant de modifier les autres variables.

Si vous avez beaucoup de mèmoire (>=256Mo) et beaucoup de tables, et que vous dèsirez des performances maximales avec un faible de nombre de clients, vous devriez essayer quelque chose cela:

```
shell> safe_mysqld -O key_buffer=64M -O table_cache=256 \
           -O sort_buffer=4M -O read_buffer_size=1M &
```

Si vous n'avez que 128Mo et seulement quelques tables, mais que vous demandez beaucoup de classements, vous pouvez essayer cela :

```
shell> safe_mysqld -O key_buffer=16M -O sort_buffer=1M
```

Si vous avez peu de mèmoire et beaucoup de connections, essayez cela:

```
shell> safe_mysqld -O key_buffer=512k -O sort_buffer=100k \
           -O read_buffer_size=100k &
```

Ou encore:

```
shell> safe_mysqld -O key_buffer=512k -O sort_buffer=16k \
           -O table_cache=32 -O read_buffer_size=8k -O net_buffer_length=1K &
```

Si vous utilisez `GROUP BY` ou `ORDER BY` sur des fichiers de taille supèrieure á la mèmoire disponible, vous devriez augmenter la valeur de `record_rnd_buffer` pour accèlèrer la lecture des lignes aprés que le classement ait ètè fait.

A l'installation de MySQL, un rèpertoire '`support-files`' est crèè, et contient plusieurs exemples de fichiers '`my.cnf`': '`my-huge.cnf`', '`my-large.cnf`', '`my-medium.cnf`' et '`my-small.cnf`'. Vous pouvez les utiliser comme base pour optimiser votre systéme.

Si vous avez vraiment beaucoup de connections, des problémes peuvent apparaître avec le fichier d'èchange si `mysqld` n'a pas ètè configurè pour utiliser peu de mèmoire pour chaque connexion. `mysqld` fonctionne mieux si vous avec suffisamment de mèmoire pour toutes les connections, bien sûr !

Notez que si vous changez une option de `mysqld`, elle ne prendra effet qu'au prochain dèmarrage du serveur.

Pour voir les effets d'un changement de paramétre, essayez quelque chose comme ca:

```
shell> mysqld -O key_buffer=32m --help
```

Assurez vous bien de la prèsence de l'option `--help` en fin de ligne; si ce n'est pas le cas, les options listèes aprés dans la ligne de commande ne seront pas prises en compte á la sortie.

5.5.3 Influences de la compilation et des liaisons sur la vitesse de MySQL

La plupart des tests suivants ont ètè rèalisès sous Linux avec les outils comparatifs de MySQL, mais ils peuvent donner quelques indications pour d'autres systémes d'exploitation et sur une charge de travail diffèrente.

Les exècutables les plus rapides sont obtenus en liant avec `-static`.

Sur Linux, le code le plus rapide sera obtenu en compilant avec `pgcc` et `-O3`. Pour compiler '`sql_yacc.cc`' avec ces options, il faut environ 200 Mo de mèmoire car `gcc/pgcc` demande beaucoup de mèmoire pour crèer toutes les fonctions d'une traite. Il est aussi possible d'utiliser `CXX=gcc` á la configuration de MySQL pour èviter l'inclusion de la bibliothéque `libstdc++` (qui n'est pas nècessaire). Sachez que pour certaines versions de `pgcc`,

le code rèsultant ne fonctionnera que sur de vrais processeurs Pentium, même si vous utilisez l'option du compilateur qui doit gènerer du code fonctionnant sur tour les types de processeurs x586 (comme AMD).

L'utilisation du meilleur compilateur et/ou de la meilleur option de compilation permet de gagner 10 á 30% de vitesse dans vos applications. C'est trés important quand vous compilez le serveur SQL vous-même !

Nous avons compilè avec les compilateurs de Cygnus CodeFusion et de Fujitsu, mais aucun des deux n'ètait suffisamment exempt d'erreurs pour permettre la compilation de MySQL avec l'optimisation.

A la compilation de MySQL, vous devriez uniquement utiliser le support des caractéres que vous allez utiliser. (Option `--with-charset=xxx`.) Les distributions binaires standards de MySQL sont compilèes avec le support de toutes les gammes de caractéres.

Voici une liste des mesures que nous avons effectuès:

- L'utilisation de `pgcc` et la compilation compléte avec l'option `-O6` donne un serveur `mysqld` 1% plus rapide qu'avec `gcc` 2.95.2.
- Si vous utilisez la liaison dynamique (sans `-static`), le rèsultat est 13% plus lent sur Linux. Sachez que vous pouvez nèanmoins utiliser la liaison dynamique pour les bibliothéques de MySQL. Seul le serveur a des performances critiques.
- Si vous allègez votre binaire `mysqld` avec l'option `strip libexec/mysqld`, vous obtenez un binaire jusqu'á 4% plus rapide.
- Si vous utilisez TCP/IP plutôt que les "sockets" Unix, le rèsultat est 7.5% plus lent sur le même ordinateur. (Si vous vous connectez sur `localhost`, MySQL utilisera les sockets par dèfaut.)
- Si vous vous connectez en TCP/IP depuis un autre ordinateur avec un lien Ethernet 100M, le rèsultat sera 8 á 11% plus lent.
- L'utilisation de connections sècurisèes (toutes les donnèes chiffrèes par le support interne de SSL) pour nos tests comparatifs a provoquè une perte de vitesse de 55%.
- Si vous compilez avec `--with-debug=full`, vous perdrez 20% de performances sur la plupart des requêtes, mais la perte peut être plus importante sur certaines requêtes (La suite de tests de MySQL tourne 35% plus lentement). Si vous utilisez `--with-debug`, vous ne perdrez que 15%. En dèmarrant une version de `mysqld`, compilèe avec `--with-debug=full`, avec with `--skip-safemalloc`, le rèsultat final devrait être proche d'une compilation avec `--with-debug`.
- Sur un Sun UltraSPARC-IIe, Forte 5.0 est 4% plus rapide que `gcc` 3.2.
- Sur un Sun UltraSPARC-IIe, Forte 5.0 est 4% plus rapide en mode 32 bit qu'en mode 64 bit.
- La compilation avec `gcc` 2.95.2 sur UltraSPARC avec l'option `-mcpu=v8 -Wa,-xarch=v8plusa` amèliore les performances de 4%.
- Sur Solaris 2.5.1, MIT-pthreads est 8-12% plus lent que la gestion native des threads de Solaris sur mono-processeur. Avec plus de charge ou de CPU, la diffèrente devrait être encore plus grande.
- Le fonctionnement avec `--log-bin` rend mysqld 1% plus lent.
- La compilation sur Linux-x86 avec gcc sans les pointeurs `-fomit-frame-pointer` ou `-fomit-frame-pointer -ffixed-ebp` rend `mysqld` 1 á 4% plus rapide.

Autrefois les distributions fournies par MySQL AB de MySQL-Linux ètaient compilèes avec `pgcc`, mais nous avons dû revenir au simple gcc á cause d'un bogue dans `pgcc` qui gènerait du code qui ne fonctionnait pas sur AMD. Nous continuerons á utiliser gcc tant que ce bogue ne sera pas corrigè. Nèanmoins, si vous avez une machine non-AMD, vous pouvez obtenir des binaires plus rapides en compilant avec `pgcc`. Le binaire standard de MySQL pour Linux est liè statiquement pour être plus rapide et plus portable.

5.5.4 Comment MySQL gére la mèmoire

La liste suivante indique certaines techniques utilisèes par le serveur `mysqld` pour gèrer la mèmoire. Lorsque c'est possible, la variable serveur lièe á la mèmoire est indiquèe :

- Le buffer de clès (variable `key_buffer_size`) est partagè par tous les threads. Les autres buffers sont sont allouès par le serveur suivant les besoins. Voir Section 5.5.2 [Server parameters], page 390.
- Chaque connexion utilise un espace spècifique au thread : une pile (par dèfaut, 64ko, variable `thread_stack`), un buffer de connexion (variable `net_buffer_length`) et un buffer de rèsultat (variable `net_buffer_length`). Le buffer de connexion et celui de rèsultat sont dynamiquement èlargit jusqu'á `max_allowed_packet` suivant les besoins. Lorsque la requête s'exècute, une copie de la chaîne de requête est aussi allouèe.
- Tous les threads partagent la même mèmoire de base.
- Seules les tables compressèes `ISAM` / `MyISAM` sont copièes en mèmoire. Ceci est dû au fait que pour un espace de 32 bits, il n'y a pas de place pour les grosses tables en mèmoire. Lorsque les systémes de 64 bits seront plus rèpandus, nous pourrions gènèraliser le support pour la copie en mèmoire.
- Chaque requête qui effectue une analyse sèquentielle d'une table, alloue un buffer de lecture (variable `record_buffer`).
- Lors de la lecture de lignes en ordre 'alèatoire' (par exemple, aprés un tri), un buffer de lecture alèatoire est allouèe pour èviter les accés disques (variable `record_rnd_buffer`).
- Toutes les jointures sont faîtes en une seule passe, et la plupart des jointure sont faîtes sans utiliser de table temporaire. La plupart des table temporaires sont faîtes en mèmoire (table `HEAP`). Les tables temporaires avec beaucoup de donnèes (calculèes comme la somme des tailles de toutes les colonnes) ou qui contiennent des colonnes de type `BLOB` sont sauvèes sur le disque.

  Un probléme avec les versions de MySQL antèrieures á la version 3.23.2 est que si une table `HEAP` dèpassait la taille maximale de `tmp_table_size`, vous obteniez une erreur `The table tbl_name is full`. Dans les nouvelles versions, ce probléme est gèrè en passant automatiquement la table `HEAP` en une table `MyISAM` sur le disque. Pour contourner ce probléme, vous pouvez augmenter la taille maximale des tables en mèmoire en modifiant l'option `tmp_table_size` de `mysqld`, ou en modifiant l'option SQL `BIG_TABLES` dans le programme client. Voir Section 5.5.6 [`SET` Syntax], page 396. En MySQL version 3.20, la taille maximale de la table temporaire est `record_buffer*16`, ce qui fait que si vous utilisez cette version, vous aurez á augmenter la valeur de `record_buffer`. Vous pouvez aussi dèmarrer `mysqld` avec l'option `--big-tables` pour toujours stocker les tables temporaires sur le disque. Cependant, cela va affecter la vitesse de votre serveur pour les requêtes complexes.

- La plupart des requêtes qui sont triées allouent un buffer de tri, et entre 0 et 2 fichiers temporaires, suivant la taille du rèsultat. Voir Section A.4.4 [Temporary files], page 701.
- Toute l'analyse et les calculs sont faits en mèmoire locale. Aucune mèmoire supplèmentaire n'est nècessaire pour les petits calculs, et les allocations et libèrations de mèmoire sont èvitès. La mèmoire n'est allouèe que pour les chaînes trés grandes (ceci se fait via `malloc()` et `free()`).
- Chaque fichier d'index est ouvert une fois, et le fichier de donnèes est ouvert pour chaque thread concurrent. Pour chaque thread concurrent, une structure de table, une structure de colonne pour chaque colonne et un buffer de taille `3 * n` est allouè (oú `n` est la taille maximale de ligne, en dehors des colonnes de type `BLOB`). Une colonne de type `BLOB` utilise 5 á 8 octets de plus que la taille des donnèes du `BLOB`. Les gestionnaires de table `ISAM`/`MyISAM` utilisent un buffer d'une ligne de plus pour leur utilisation interne.
- Pour chaque table qui a une colonne `BLOB`, un buffer est dynamiquement agrandi pour lire les valeurs `BLOB`. Si vous analysez toute une table, un buffer aussi grand que la plus grande valeur de la colonne `BLOB` sera allouè.
- Les gestionnaires de tables pour les tables en cours d'utilisation sont sauvèes dans un cache, et gèrè comme une pile FIFO. Normalement, ce cache contient 64 lignes. Si une table doit être utilisèe par deux threads concurrents simultanèment, le cache contiendra deux entrèes pour la table. Voir Section 5.4.7 [Table cache], page 388.
- La commande `mysqladmin flush-tables` ferme toute les tables qui ne sont pas utilisèes, et marque toutes les tables en cours d'utilisation pour qu'elles soient fermèes dés la fin du thread. Cela va libèrer l'essentiel de la mèmoire utilisèe.

`ps` et d'autres commandes de statut systéme peuvent indiquer que `mysqld` utilise beaucoup de mèmoire. Ceci est peut être dû a des erreurs de comptabilitè. Par exemple, sous Solaris, `ps` compte la mèmoire inutilisèe entre les threads comme de la mèmoire utilisèe. Vous pouvez le vèrifier en regardant l'ètat de la swap avec `swap -s`. Nous avons testè `mysqld` avec les dètecteurs de fuite mèmoire commerciaux, et il n'y a aucune fuite.

5.5.5 Comment MySQL utilise le DNS

Quand un nouveau thread se connecte á `mysqld`, `mysqld` crèe nouveau thread pour traiter la requête. Ce thread contrôle d'abord si le nom de l'hôte est dans le cache des noms d'hôte. Si ce n'est pas le cas, le thread va appeler `gethostbyaddr_r()` et `gethostbyname_r()` pour rèsoudre le nom de l'hôte.

Si le systéme d'exploitation ne supporte pas les appels prècèdents, le thread va verrouiller un "mutex" et appeler `gethostbyaddr()` et `gethostbyname()` á la place. Sachez que dans ce cas, aucun autre thread ne peut rèsoudre de nom d'hôte qui n'est pas dans le cache tant que le premier thread n'a pas fini.

Il est possible de dèsactiver la recherche du nom par DNS en dèmarrant `mysqld` avec l'option `--skip-name-resolve`. Dans ce cas, il est toujours possible d'utiliser les adresses IP dans les tables de priviléges de MySQL.

Si votre service DNS est trés lent et que vous avez beaucoup d'hôtes, vous pouvez amèliorer les performances soit en dèsactivant le DNS avec `--skip-name-resolve`, soit en augmentant la taille de `HOST_CACHE_SIZE` (par dèfaut: 128) et en recompilant `mysqld`.

Il est possible de dèsactiver le cache de noms d'hôte avec `--skip-host-cache`. Il est possible de vider le cache des noms d'hôtes avec `FLUSH HOSTS` ou avec `mysqladmin flush-hosts`.

Si vous ne voulez pas autoriser les connections par `TCP/IP`, vous pouvez utiliser l'option `--skip-networking` au dèmarrage de `mysqld`.

5.5.6 Syntaxe de `SET`

```
SET [GLOBAL | SESSION] variable_sql=expression, [[GLOBAL | SESSION] variable_sql=ex
```

`SET` permet de configurer plusieurs options qui affectent le comportement de votre serveur ou de votre client.

Les exemples suivants montrent les diffèrentes syntaxes qu'on peut utiliser pour configurer des variables :

Dans les anciennes versions de MySQL nous avons permis l'utilisation de la syntaxe `SET OPTION`, mais celle-ci est á prèsent dèsapprouvèe.

Dans la version 4.0.3 de MySQL nous avons ajoutè les options `GLOBAL` et `SESSION` et l'accés aux variables de configuration les plus importantes.

`LOCAL` peut être utilisè en tant que synonyme de `SESSION`.

Si vous configurez plusieurs variables sur une seule ligne de commande, le dernier mode `GLOBAL | SESSION` utilisè est pris en compte.

```
SET sort_buffer_size=10000;
SET @@local.sort_buffer_size=10000;
SET GLOBAL sort_buffer_size=1000000, SESSION sort_buffer_size=1000000;
SET @@sort_buffer_size=1000000;
SET @@global.sort_buffer_size=1000000, @@local.sort_buffer_size=1000000;
```

La syntaxe `@@nom_variable` est supportèe pour rendre MySQL compatible avec d'autres bases de donnèes.

Le diffèrentes variables systéme qu'on peut configurer sont dècrites dans la section variables systéme de ce manuel. Voir Section 6.1.5 [System Variables], page 409.

Si vous utilisez `SESSION` (par dèfaut) l'option que vous configurez garde son effet jusqu'á ce que la session courante se termine, ou que vous modifiez á nouveau cette option. Si vous utilisez `GLOBAL`, qui requiére le privilége `SUPER`, l'option est gardèe en mèmoire et utilisèe pour les nouvelles connexion jusqu'au redèmarrage du serveur. Si vous voulez qu'un changement reste permanent, vous devez l'effectuer dans l'un des fichiers d'options de MySQL. Voir Section 4.1.2 [Option files], page 198.

Pour èviter un mauvais usage, MySQL donnera une erreur si vous utilisez `SET GLOBAL` avec une variable qui ne peut être inutilisèe que par `SET SESSION` ou si vous n'utilisez pas `SET GLOBAL` avec une variable globale.

Si vous voulez configurer une variable `SESSION` á une valeur `GLOBAL` ou une valeur `GLOBAL` á la valeur par dèfaut de MySQL, vous pouvez la configurer á `DEFAULT`.

```
SET max_join_size=DEFAULT;
```

Ceci est identique á :

```
SET @@session.max_join_size=@@global.max_join_size;
```

Si vous voulez poser une limite sur la valeur maximale qu'une valeur de variable peut prendre avec la commande `SET`, vous pouvez le spècifier en utilisant l'option de ligne de commande `--maximum-nom-variable`. Voir Section 4.1.1 [Command-line options], page 192.

Vous pouvez obtenir une liste de la plupart des variables avec `SHOW VARIABLES`. Voir Section 4.5.6.4 [SHOW VARIABLES], page 273. Vous pouvez obtenir la valeur d'une variable spècifique avec la syntaxe `@@[global.|local.]nom_variable` :

```
SHOW VARIABLES like "max_join_size";
SHOW GLOBAL VARIABLES like "max_join_size";
SELECT @@max_join_size, @@global.max_join_size;
```

Vous trouverez ici une description des variables qui utilisent une syntaxe non-standard de `SET`. Les dèfinitions des autres variables peuvent être trouvèes dans la section des variables systéme, avec les options de dèmarrage ou dans la description de `SHOW VARIABLES`. Voir Section 6.1.5 [System Variables], page 409. Voir Section 4.1.1 [Command-line options], page 192. Voir Section 4.5.6.4 [SHOW VARIABLES], page 273.

`CHARACTER SET nom_jeu_de_caractéres DEFAULT`
: Cela change le jeu de caractére dans toutes les chaînes du et vers le client avec le jeu donnè. Jusqu'á maintenant, la seule option pour `nom_jeu_de_caractéres` est `cp1251_koi8`, mais vous pouvez facilement ajouter d'autres possibilitès en èditant le fichier '`sql/convert.cc`' dans la distribution des sources MySQL. Le jeu de caractéres par dèfaut peut être restaurè en utilisant la valeur `DEFAULT` de `nom_jeu_de_caractéres DEFAULT`.

  Notez que la syntaxe pour configurer l'option `CHARACTER SET` différe de la syntaxe pour configurer les autres options.

`PASSWORD = PASSWORD('un mot de passe')`
: Change le mot de passe pour l'utilisateur courant. Chaque utilisateur non anonyme peut changer son propre mot de passe !

`PASSWORD FOR utilisateur = PASSWORD('un mot de passe')`
: Configure un mot de passe pour un utilisateur spècifique sur le serveur courant. Seul un utilisateur avec l'accés á la base de donnèes `mysql` peut faire cela. L'utilisateur doit être donnè au format `utilisateur@hote`, oú `utilsateur` et `hote` sont exactement comme ils sont listès dans les colonnes `User` et `Host` des entrèes de la table `mysql.user`. Par exemple, si vous avez une entrèe avec dans les champs `User` et `Host` les entrèes `'bob'` et `'%.loc.gov'`, vous ècrirez :

  ```
  mysql> SET PASSWORD FOR bob@"%.loc.gov" = PASSWORD("nouveaupass");
  ```

  Ce qui est èquivalent á :

  ```
  mysql> UPDATE mysql.user SET password=PASSWORD("nouveaupass")
      ->                   WHERE user="bob' AND host="%.loc.gov";
  ```

`SQL_AUTO_IS_NULL = 0 | 1`
: Si dèfinie á `1` (par dèfaut) alors on peut trouver la derniére ligne insèrèe dans une table avec une colonne `AUTO_INCREMENT` avec la construction suivante : `WHERE auto_increment_column IS NULL`. Ceci est utilisè par des programmes ODBC tel que Access.

`AUTOCOMMIT= 0 | 1`

Si dèfinie á `1` tous les changements dans une table se feront en une seule fois. Pour dèmarrer une transaction multi-commandes, vous devez utiliser la commande `BEGIN`. Voir Section 6.7.1 [COMMIT], page 518. Si dèfinie á `0` vous devez utiliser `COMMIT` / `ROLLBACK` pour accepter/annuler cette transaction. Voir Section 6.7.1 [COMMIT], page 518. Notez que quand vous passez du mode non `AUTOCOMMIT` vers le mode `AUTOCOMMIT`, MySQL fera un `COMMIT` automatique sur toutes les transactions en cours.

`BIG_TABLES = 0 | 1`

Si dèfinie á `1`, toutes les tables temporaires sont stockèes sur le disque plutôt qu'en mèmoire. Cela sera un peu plus lent, mais vous n'obtiendrez jamais l'erreur `The table nom_de_table is full` pour les grands `SELECT` qui requiérent une table temporaire. La valeur par dèfaut pour une nouvelle connexion est `0` (qui est d'utiliser la mèmoire pour les tables temporaires). Cette option se nommait avant `SQL_BIG_TABLES`.

`SQL_BIG_SELECTS = 0 | 1`

Si configurè á `0`, MySQL interrompra les requêtes `SELECT` qui prendront probablement trop de temps. C'est utile lorsqu'une clause `WHERE` dèconseillèe a ètè utilisèe. Une grosse requête est dèfinie comme ètant un `SELECT` qui devra probablement ètudier plus de `max_join_size` lignes. La valeur par dèfaut d'une nouvelle connexion est `1` (qui permet toutes les requêtes `SELECT`).

`SQL_BUFFER_RESULT = 0 | 1`

`SQL_BUFFER_RESULT` forcera les rèsultats des requêtes `SELECT` á être placès dans une table temporaire. Cela aidera MySQL á libèrer les verrous sur table plus tôt et amèliorera les cas oú le jeu de rèsultats de la requête prend trop de temps á être envoyèe au client.

`LOW_PRIORITY_UPDATES = 0 | 1`

Si vous configurez á `1`, toutes les commandes `INSERT`, `UPDATE`, `DELETE`, et `LOCK TABLE WRITE` attendent qu'il n'y ai plus de `SELECT` ou `LOCK TABLE READ` sur la table affectèe. Cette option s'appelait avant `SQL_LOW_PRIORITY_UPDATES`.

`MAX_JOIN_SIZE = valeur | DEFAULT`

Ne pas permettre les requêtes `SELECT` qui auront besoin d'examiner plus de `valeur` combinaisons de lignes. En configurant cette valeur, vous pouvez capturer les requêtes `SELECT` oú les clefs sont mal utilisèes et qui prendront probablement beaucoup de temps á s'exècuter. Prendre une autre valeur que `DEFAULT` mettre á zèro l'option `SQL_BIG_SELECTS`. Si vous reconfigurez l'option `SQL_BIG_SELECTS`, la variable `SQL_MAX_JOIN_SIZE` sera ignorèe. Vous pouvez configurer une valeur par dèfaut pour cette variable en dèmarrant `mysqld` avec `-O max_join_size=#`. Cette option s'appelait avant `SQL_MAX_JOIN_SIZE`.

Notez que si le rèsultat de la requête est dèjá dans le cache de requêtes, le test prècèdent ne sera pas effectuè. A la place, MySQL renverra le rèsultat au client. Puisque le rèsultat de la requête est dèjá traitè et qu'il sera aisè au serveur d'envoyer la rèponse au client.

`QUERY_CACHE_TYPE = OFF | ON | DEMAND`
`QUERY_CACHE_TYPE = 0 | 1 | 2`
: Configure le cache de requêtes pour ce processus.

Option	Description
0 ou OFF	Ne pas cacher ou rècupèrer les rèsultats.
1 ou ON	Met tous les rèsultats en cache á part les requêtes `SELECT SQL_NO_CACHE ...`.
2 ou DEMAND	Ne met en cache que les requêtes `SELECT SQL_CACHE ...`.

`SQL_SAFE_UPDATES = 0 | 1`
: Si dèfinit á `1`, MySQL annulera si un `UPDATE` ou un `DELETE` est exècutè alors qu'il n'utilise pas de clef ou de `LIMIT` dans la clause `WHERE`. Cela permet de bloquer les requêtes erronèes crèes á la main.

`SQL_SELECT_LIMIT = valeur | DEFAULT`
: Le nombre maximal des enregistrements que doivent retourner les requêtes `SELECT`. Si un `SELECT` posséde une clause `LIMIT`, celle-ci est utilisèe. La valeur par dèfaut pour une nouvelle connexion est "illimitèe." Si vous avez changè la limite, la valeur par dèfaut peut être retrouvèe en utilisant la valeur `DEFAULT` avec `SQL_SELECT_LIMIT`.

`SQL_LOG_OFF = 0 | 1`
: Si dèfinie á `1`, aucune entrèe ne sera ècrite dans le log standard pour ce client, si le client á le privilége `SUPER`. Cela n'affecte pas le log des mises á jours !

`SQL_LOG_UPDATE = 0 | 1`
: Si dèfinie á `0`, aucune trace des requêtes ne sera gardèe dans le log des mises á jour pour le client, si le client á le privilége `SUPER`. Cela n'affecte pas le log standard !

`SQL_QUOTE_SHOW_CREATE = 0 | 1`
: Si vous le configurez á `1`, `SHOW CREATE TABLE` protègera les noms de tables et de colonnes. Ceci est **activè** par dèfaut, pour que la rèplication des tables avec des noms á risques fonctionne. Section 4.5.6.8 [`SHOW CREATE TABLE`], page 286.

`TIMESTAMP = valeur_timestamp | DEFAULT`
: Configure le temps pour ce client. C'est utilisè pour obtenir le timestamp d'origine si vous utilisez le log de mises á jour pour restaurer des lignes. `valeur_timestamp` doit être un timestamp Unix, et non un timestamp MySQL.

`LAST_INSERT_ID = #`
: Configure la valeur qui doit être retournèe par `LAST_INSERT_ID()`. C'est enregistrè dans le log de mises á jour quand vous utilisez `LAST_INSERT_ID()` dans une commande qui met á jour une table.

`INSERT_ID = #`
: Configure la valeur á utiliser par l'appel suivant á la commande `INSERT` ou `ALTER TABLE` lors de l'insertion d'une valeur `AUTO_INCREMENT`. Cela est souvent utilisè par le log des mises á jour.

5.6 Problémes avec les disques

- Comme mentionnè plus tôt, les accés disques reprèsentent une limitation. Ce probléme devient de plus en plus apparent, au fur et á mesure que les donnèes sont de plus en plus nombreuses, et que les techniques de cache deviennent impossibles. Pour les grandes bases de donnèes, lorsque vous accèdez aux donnèes plus ou moins alèatoirement, vous pouvez être sûr que vous aurez besoin d'un accés disque pour lire, et de plusieurs autres pour ècrire. Pour minimiser le probléme, utilisez des disques avec des temps d'accés trés faibles.
- Augmentez le nombre de disque disponibles (et donc, rèduisez le coût d' un accés), en plaçant des donnèes sur d'autres fichiers via des liens symboliques.

  Utiliser des liens symboliques
  : Cela signifie que vous allez faire un lien symbolique sur le fichier d'index et/ou le fichier de donnèes sur un autre disque. Cela amèliore les lectures et ècriture (surtout si ces disques ne sont alors utilisès qu'á ça). Voir Section 5.6.1 [Symbolic links], page 401.

  Paralellisme
  : Le parallèlisme signifie que vous avez plusieurs disques matèriel, et que vous ècrivez le premier bloc de donnèes sur le premier disque, puis le second bloc de donnèes sur le second disque, et le n-iéme bloc sur le n-iéme disque, etc. Cela signifie que si la taille normale de vos donnèes sont moins grand que le nombre de disque disponibles, vous obtiendrez alors des performances additionnèes. Notez que le parallèlisme est trés dèpendant du nombre de disque disponibles et du systéme d'exploitation. Voir Section 5.1.5 [Custom Benchmarks], page 361.

    Notez que la diffèrence de performance avec le parallèlisme est **trés** dèpendante des paramétres. Suivant la façon avec laquelle vous avez configurè les disques en paralléle, et le nombre de disque que vous utilisez, le facteur d'amèlioration peut être trés variable. Notez que vous devez faire votre optimisation en lecture alèatoire ou sèquentielle.
- Pour plus de confiance, vous pouvez utiliser des disques en RAID 0+1 (parallèlisme et rèplication), mais dans ce cas, vous aurez besoin de 2*N disques pour contenir vos donnèes sur N disques. C'est probablement l'option la plus sûre, si vous avez le budget pour cela. Vous risquez aussi d'avoir á investir dans un systéme de gestion de gros volume de donnèes pour gèrer cela efficacement.
- Une bonne option est de garder les donnèes semi-importantes (qui peuvent être regènèrèes) sur un disque RAID 0 tandis que les donnèes vraiment importantes (comme les informations d'hôtes et les log) sur un disque de type RAID 0+1 ou RAID N. RAID N peut être un probléme si vous avez de nombreux accés en ècrire, á cause du temps de modification des bits de paritè.
- Vous pouvez aussi optimiser les paramétres du systéme de fichier que la base de donnèes utilise. Une modification simple est de monter le systéme de fichier avec l'option noatime. Cela fait que le systéme ne fait plus la modification de la date de dernier accés á l'inode, ce qui èvite des accés disques.

- Sous Linux, vous pouvez amèliorer les performances (jusqu'á 100% en charge n'est pas difficile) en utilisant `hdparm` pour configurer votre interface disque. La commande suivante doit être une sèrie de bonnes options de `hdparm` pour MySQL (et probablement d'autres applications) :

  ```
  hdparm -m 16 -d 1
  ```

  Notez que la performances et la robustesse des solutions ci-dessus dèpendent de votre matèriel, et nous vous conseillons vivement de tester votre systéme soigneusement aprés avoir utilisè `hdparm`! Consultez le manuel de `hdparm` pour plus de dètails. Si `hdparm` n'est pas utilisè correctement, le systéme de fichiers peut être corrompu. Sauvegardez tout avant d'expèrimenter.
- Sur de nombreux systémes d'exploitation, vous pouvez monter des disques avec l'option `-o async` pour que le systéme de fichiers soit modifiè de maniére asynchrone. Si votre serveur est raisonnablement stable, vous devriez obtenir de bonne performances sans sacrifier la stabilitè (cette option est activèe par dèfaut sur Linux).
- Si vous n'avez pas besoin de savoir quand un fichier a ètè accèdè la derniére fois (ce qui n'est pas utile avec un serveur de base de donnèes), vous pouvez monter votre systéme de fichier avec l'option `-o noatime`.

5.6.1 Utiliser des liens symboliques

Vous pouvez dèplacer les dossiers de bases de donnèes et les placer dans un autre endroit, puis remplacer les dossiers eux-mêmes par des liens symboliques vers ces autres endroits. Vous pourriez vouloir faire cela pour mettre la base de donnèes sur un systéme de fichier plus rapide, ou pour gagner de l'espace disque sur le systéme central, ou encore rèpartir vos tables sur diffèrents disques.

Le mieux, pour cela, est de faire des liens symboliques des bases vers les diffèrents disques, et de ne faire des liens symboliques sur les tables qu'en dernier ressort.

5.6.1.1 Utiliser les liens symboliques pour les bases

Pour crèer des liens symboliques sur les bases de donnèes, vous devez commencer par crèer un dossier sur un disque de destination, puis faire un lien symbolique depuis le dossier de donnèes vers votre dossier de destination.

```
shell> mkdir /dr1/databases/test
shell> ln -s /dr1/databases/test mysqld-datadir
```

MySQL n'accepte pas que vous fassiez le lien depuis plusieurs bases sur le même dossier. Remplacer une base par un lien symbolique sera correct tant que vous n'essayez pas de faire des liens symboliques dans la même base. Supposez que vous la base `db1` dans le dossier de donnèes MySQL, puis que vous fassiez un lien symbolique `db2` qui pointe sur `db1` :

```
shell> cd /path/to/datadir
shell> ln -s db1 db2
```

Maintenant, pour toute table `tbl_a` de `db1`, il en apparaît aussi `tbl_a` dans `db2`. Si un thead modifie `db1.tbl_a` et un autre `db2.tbl_a`, il va y avoir un conflit.

Si vous avez vraiment besoin de cette fonctionnalitè, vous devez changer le code suivant dans le fichier C '`mysys/mf_format.c`':

```
if (flag & 32 || (!lstat(to,&stat_buff) && S_ISLNK(stat_buff.st_mode)))
```

en

```
if (1)
```

Sous Windows, vous pouvez utiliser des liens internes symboliques pour relier des bases en compilant MySQL avec l'option `-DUSE_SYMDIR`. Cela vous permettra de placer vos bases de donnèes sur diffèrentes partitions. Voir Section 2.6.2.5 [Windows symbolic links], page 125.

5.6.1.2 Utiliser des liens symboliques avec les tables

Avant MySQL 4.0, vous ne devez pas utiliser les liens symboliques avec les tables, si vous n'êtes pas trés prudents avec. Le probléme est que si vous exècutez `ALTER TABLE`, `REPAIR TABLE` ou `OPTIMIZE TABLE` sur une table symbolique, le lien sera supprimè et remplacè par le fichier original. Cela arrive car les commandes ci-dessus fonctionnent en crèant un fichier temporaire dans le dossier de base, et lorsque l'opèration est faite, l'original est remplacè par la copie.

Vous ne devez pas utiliser des liens symboliques sur les tables, sur les systémes qui ne supportent pas complétement la fonction `realpath()`. (Au moins Linux et Solaris supportent `realpath()`)

En MySQL 4.0, les liens symboliques sont complétement supportès par les tables `MyISAM`. Les autres types de tables vous donneront des rèsultats ètranges lorsque vous les utilisez comme indiquè ci-dessus.

La gestion des liens symboliques de MySQL 4.0 fonctionne comme ceci (uniquement pour les tables `MyISAM`) :

- Dans le dossier de donnèes, vous allez toujours trouver le fichier de dèfinition de table, le fichier de structure et le fichier d'index.
- Vous pouvez utiliser un lien symbolique avec le fichier d'index et celui de donnèes, pour placer ces fichiers dans d'autres dossiers.
- Le lien symbolique peut être fait via le systéme d'exploitation (si `mysqld` ne fonctionne pas) ou avec la commande `INDEX/DATA DIRECTORY="path-to-dir"` dans `CREATE TABLE`. Voir Section 6.5.3 [CREATE TABLE], page 504.
- `myisamchk` ne va pas remplacer un lien symbolique avec les donnèes ou le fichier d'index, mais il va travailler directement sur le fichier vers lequel le lien pointe. Tous les fichiers temporaires seront crèè dans le même dossier que le dossier qui contient les donnèes ou le fichier d'index.
- Lorsque vous dètruisez une table qui utilise un lien symbolique, le fichier et le lien symbolique sont dètruits. C'est une bonne raison pour **ne pas** exècuter `mysqld` en tant que `root` ou donner des droits d'ècriture á d'autres personnes dans les dossiers de donnèes de MySQL.
- Si vous renommez une table avec `ALTER TABLE RENAME` vous n'avez pas á dèplacer la table dans une autre base, le lien symbolique du dossier de base sera renommè avec le nouveau nom.
- Si vous utilisez la commande `ALTER TABLE RENAME` pour dèplacer la table dans une autre base, la table sera dèplacèe dans l'autre base, et l'ancien lien symbolique et le

fichier vers lequel il pointait seront dètruits (en d'autres termes, la nouvelle table ne sera pas un lien symbolique).

- Si vous n'utilisez pas de lien symbolique, vous devriez utiliser l'option `--skip-symlink` de `mysqld` pour vous assurer que personne n'efface ou ne renomme un fichier en dehors du dossier de données de MySQL.

Ce qui n'est pas encore supportè :

- `ALTER TABLE` ignore toutes les options `INDEX/DATA DIRECTORY="path"`.
- `CREATE TABLE` n'indique pas si la table a des liens symboliques.
- `mysqldump` n'inclut pas d'informations sur les informations de liens.
- `BACKUP TABLE` et `RESTORE TABLE` ne respectent pas les liens symboliques.

6 Rèfèrence du langage MySQL

MySQL posséde une interface SQL assez complexe, mais facile á apprendre. Ca chapitre dècrit les diffèrentes commandes, types, et fonctions que vous aurez besoin de connaître pour utiliser MySQL de façon optimale. Ce chapitre sert aussi de rèfèrence pour toutes les fonctionnalitès incluses au sein de MySQL. Afin d'utiliser ce chapitre efficacement, vous pourrez trouver utile de vous reporter aux diffèrents index.

6.1 Structure du langage

6.1.1 Literals: Comment ècrire les chaînes et les nombres

Cette section dècrit les diffèrents façons d'ècrire les chaînes et les nombres en MySQL. Elle couvre aussi les diffèrentes nuances et quiproquos que vous pouvez rencontrer lorsque vous manipulez ces types de donnèes.

6.1.1.1 Chaînes

Une chaîne est une sèquence de caractéres, entourèe de guillemets simples (‘’’) ou doubles (‘"’) (simple seulement si vous êtes en mode ANSI). Exemples:

```
'une chaîne'
"une autre chaîne"
```

A l'intèrieur d'une chaîne, certains sèquences de caractéres ont une signification spèciale. Chacune d'elle commence par un anti-slash (‘\’), connu comme le *caractére d'èchappement*. `MySQL` reconnat les sèquences suivantes :

`\0` Un 0 ASCII (`NUL`).

`\'` Un guillemet simple (‘’’).

`\"` Un guillemet double (‘"’).

`\b` Un effacement.

`\n` Une nouvelle ligne.

`\r` Un retour chariot.

`\t` Une tabulation.

`\z` ASCII(26) (Contrle-Z). Ce caractére peut être encodè pour vous èviter des problémes avec Windows, vu qu'il èquivaut á une fin de fichier sur cet OS. (ASCII(26) vous posera des problémes si vous utilisez `mysql base < fichier`.)

`\\` Un anti-slash (‘\’).

`\%` Un ‘%’. Cette notation est utilisèe pour trouver les occurrences de ‘%’ au cas oú ‘%’ peut aussi être interprètè comme un caractére spècial. Voir Section 6.3.2.1 [String comparison functions], page 452.

`\_` Un '_'. Cette notation est utilisèe pour trouver les occurrences de '_' au cas oú '_' peut aussi être interprètè comme un caractére spècial. Voir Section 6.3.2.1 [String comparison functions], page 452.

Notez que si vous utilisez '`\%`' ou '`\_`' dans un contexte de chaînes de caractéres, ces sèquences retourneront '`\%`' et '`\_`' et non '`%`' et '`_`'.

Il y a plusieurs façons d'intgrer un guillemet dans une chaîne :

- Un '`''`' á l'intèrieur d'une chaîne entourèe de '`''`' peut être notè '`''''`'.
- Un '`"`' á l'intèrieur d'une chaîne entourèe de '`"`' peut être notè '`""`'.
- Vous pouvez faire prècèder le guillemet par caractére d'èchappement ('`\`').
- Un guillemet simple '`''`' á l'intèrieur d'une chaîne á guillemets doubles '`"`' n'a besoin d'aucun traitement spècial (ni doublage, ni èchappement). De même, aucun traitement spècial n'est requis pour un guillemet double '`"`' á l'intèrieur d'une chaîne á guillemets simples '`''`'.

Le `SELECT` montrè ici explique comment les guillemets et les èchappements fonctionnent :

```
mysql> SELECT 'bonjour', '"bonjour"', '""bonjour""', 'bon''jour', '\'bonjour';
+---------+-----------+-------------+----------+----------+
| bonjour | "bonjour" | ""bonjour"" | bon'jour | 'bonjour |
+---------+-----------+-------------+----------+----------+

mysql> SELECT "bonjour", "'bonjour'", "''bonjour''", "bon""jour", "\"bonjour";
+---------+-----------+-------------+----------+----------+
| bonjour | 'bonjour' | ''bonjour'' | bon"jour | "bonjour |
+---------+-----------+-------------+----------+----------+

mysql> SELECT "Voilá\n3\nlignes";
+--------------------+
| Voilá
3
lignes |
+--------------------+
```

Si vous voulez insèrer des donnèes binaires dans un champ chaîne (comme un `BLOB`), les caractéres suivants doivent être èchappès :

`NUL` ASCII 0. Reprèsentez le avec '`\0`' (un anti-slash suivi du caractére ASCII '`0`').

`\` ASCII 92, anti-slash. A reprèsenter avec '`\\`'.

`'` ASCII 39, guillemet simple. A reprèsenter avec '`\'`'.

`"` ASCII 34, guillemet double. A reprèsenter avec '`\"`'.

Si vous ècrivez du code C, vous pouvez utiliser la fonction `mysql_real_escape_string()`, fournie par l'API, pour èchapper les caractéres pour les requêtes `INSERT`. Voir Section 8.4.2 [C API function overview], page 612. Avec Perl, vous pouvez utiliser la mèthode `quote` du package `DBI` pour convertir les caractéres spèciaux en leurs èquivalents èchappès. Voir Section 8.2.2 [Perl `DBI` Class], page 592.

Vous devez utiliser une fonction d'èchappement pour chaque chaîne susceptible de contenir l'un des caractéres spèciaux listès plus haut.

Alternativement, beaucoup d'API `MySQL`, fournissent des fonctionnalitès qui vous permettent d'insèrer des marqueurs spèciaux dans une requête puis de les remplacer par des valeurs lors de l'exècution de celle-ci. Dans ce cas, l'API s'occupe d'èchapper les caractéres spèciaux pour vous.

6.1.1.2 Nombres

Les entiers sont reprèsentès comme une sèquence de chiffres. Les dècimaux utilisent '`.`' comme sèparateur. Tous les types de nombres peuvent être prècèdès d'un '`-`' pour indiquer une valeur nègative.

Exemples d'entiers valides :

```
1221
0
-32
```

Exemples de nombres á virgule flottante :

```
294.42
-32032.6809e+10
148.00
```

Un entier peut être utilisè dans un contexte dècimal, il sera interprètè comme le nombre dècimal èquivalent.

6.1.1.3 Valeurs hexadècimales

MySQL supporte les valeurs hexadècimales. Dans un contexte numèrique, elles agissent comme des entiers (prècision 64-bit). Dans un contexte de chaîne, elles agissent comme une chaîne binaire oú chaque paire de caractéres hexadècimaux est convertie en caractére :

```
mysql> SELECT x'4D7953514C';
         -> MySQL
mysql> SELECT 0xa+0;
         -> 10
mysql> SELECT 0x5061756c;
         -> Paul
```

La syntaxe `x'chaînehexa'` (nouvelle en 4.0) est basèe sur ANSI SQL et la syntaxe `0x` est basèe sur ODBC. Les chaînes hexadècimales sont souvent utilisèe par ODBC pour fournir des valeurs aux colonnes `BLOB`. Vous pouvez convertir une chaîne ou un nombre en hexadècimal avec la fonction `HEX()`.

6.1.1.4 Valeurs `NULL`

La valeur `NULL` signifie "pas de donnèes" et est diffèrente des valeurs comme `0` pour les nombres ou la chaîne vide pour les types chaîne. Voir Section A.5.3 [Problems with `NULL`], page 704.

`NULL` peut être reprèsentè par `\N` lors de la rècupèration ou ècriture avec des fichiers (`LOAD DATA INFILE`, `SELECT ... INTO OUTFILE`). Voir Section 6.4.9 [LOAD DATA], page 497.

6.1.2 Noms de bases, tables, index, colonnes et alias

Les noms des bases de donnèes, tables, index, colonnes et alias suivent tous les mêmes régles en MySQL.

Notez que les régles ont changè á partir de la version 3.23.6 de MySQL lorsqu'á ètè introduite la protection des noms d'identifiant avec les guillemets obliques ‘`. "` fonctionne aussi si vous fonctionnez en mode AINSI. Voir Section 1.7.2 [ANSI mode], page 33.

Identifiant	Longueur maximale	Caractéres autorisès
Base de donnèes	64	Tous les caractéres autorisès dans un nom de dossier á part ‘/’, ‘\’ et ‘.’.
Table	64	Tous les caractéres autorisès dans le nom d'un fichier á part ‘/’ et ‘.’.
Colonne	64	Tous.
Alias	255	tous.

Notez qu'en plus de ce qui prècéde, vous n'avez pas droit au caractéres ASCII(0), ASCII(255) ou aux guillemets dans un identifiant.

Notez que si un identifiant est un mot rèservè, ou contient des caractéres spèciaux, vous devez absolument le protèger avec ‘ :

```
mysql> SELECT * FROM ‘select‘ WHERE ‘select‘.id > 100;
```

Voir Section 6.1.7 [Reserved words], page 414.

Dans les versions antèrieures á la 3.23.6, les régles ètaient les suivantes :

- Un nom est constituè d'une sèquence de caractéres alpha-numèriques, tirè du jeu de caractéres courant, et en y ajoutant le soulignè ‘_’ et le signe dollar ‘$’. Le jeu de caractéres par dèfaut est ISO-8859-1 Latin1; Il peut être modifiè avec l'option de dèmarrage `--default-character-set` de `mysqld`. Voir Section 4.6.1 [Character sets], page 286.
- Un nom peut commencer par n'importe quel caractére permis dans un nom. Un nom peut donc commencer par un chiffre. (Ce qui n'est pas le cas dans d'autres systémes de base de donnèes). Par contre, un nom ne peut consister *uniquement* de chiffres.
- Vous ne pouvez pas utiliser de point (‘`.`’) dans les noms car ce caractére est utilisè pour se rèfèrer au colonnes (voir ce qui suit).

Il est conseillè de ne pas utiliser de nom comme `1e`, car une expression comme `1e + 1` est ambigue. Ce nom pourra être interprètè comme l'expression `1e + 1` ou le nombre `1e+1`.

Avec MySQL vous pouvez indiquer une colonne avec l'une des syntaxes suivantes :

Rèfèrence	Signification
`nom_colonne`	La colonne `nom_colonne` de chaque table utilisèe dans la requête qui comporte une telle colonne.
`nom_de_table.nom_colonne`	La colonne `nom_colonne` de la table `nom_de_table` de la base de donnèes courante.
`nom_de_base.nom_de_table.nom_colonne`	La colonne `nom_colonne` de la table `nom_de_table` de la base `nom_de_base`. Cette forme est disponible á partir de la version 3.22 de MySQL.

`'nom_colonne'` Une colonne qui est un mot rèservè ou qui contient un caractére spècial.

Vous n'avez pas besoin de spècifier `nom_de_table` ou `nom_de_base.nom_de_table` avant une colonne sauf si la rèfèrence risque d'être ambigue. Par exemple, supposons que `t1` et `t2` contiennent toutes les deux une colonne nommèe `c`, et que vous lisez `c` dans une requête `SELECT` qui utilise `t1` et `t2` á la fois. Dans ce cas, `c` est ambigu car il peut se rèfèrer `t1` aussi bien qu'á `t2`. Vous devez donc indiquer la table ciblèe en ècrivant `t1.c` ou `t2.c`. De même, si vous lisez á partir d'une table `t` situèe dans la base `db1` et de la table `t` dans la base `db2`, vous devez vous rèfèrer aux colonnes de ces tables en ècrivant `db1.t.nom_colonne` et `db2.t.nom_colonne`.

La syntaxe `.nom_de_table` rèfèrence la table `nom_de_table` dans la base de donnèes courante. Cette syntaxe est acceptèe pour assurer la compatibilitè avec ODBC, car certains pilotes ODBC prèfixent les noms des tables avec des points ('`.`').

6.1.3 Sensibilitè á la casse pour les noms

En MySQL, les bases et les tables correspondent á des dossiers et des fichiers. Par consèquent, la sensibilitè á la casse du systéme dètèrminera la sensisbilitè á la casse des noms de bases de donnèes et tables. Cela signifie que les noms sont insensibles á la casse sous Windows, et sensibles sous la plupart des variètès Unix (Mac OS X ètant une exception). Voir Section 1.7.3 [Extensions to ANSI], page 34.

Note : Même si les noms ne sont pas sensibles á la casse sous Windows, vous ne devez pas vous rèfèrer á une entitè en utilisant diffèrentes casse dans la même requête. La requête suivante ne fonctionnera pas car elle se rèfère á une table avec `ma_table` et `MA_TABLE`:

```
mysql> SELECT * FROM ma_table WHERE MA_TABLE.col=1;
```

Les noms de colonnes et d'alias sont insensibles á la casse dans tous les cas.

Les alias sur tables sont sensibles á la casse. La requête suivante ne marchera pas car elle se rèfére á `a` et `A` :

```
mysql> SELECT nom_de_colonne FROM nom_de_table AS a
    -> WHERE a.nom_de_colonne = 1 OR A.nom_de_colonne = 2;
```

Si vous avez du mal á vous souvenir de la casse des noms de bases et de tables, adoptez une convention, comme toujours crèer les bases et les tables en utilisant des minuscules.

Un moyen d'èviter ce probléme est de dèmarrer `mysqld` avec `-O lower_case_table_names=1`. Par dèfaut, cette option vaut 1 sur Windows et 0 sur Unix.

Si `lower_case_table_names` est á 1 MySQL convertira tout les noms de tables en minuscules lors des sauvegardes et des rècupèrations. (A partir de la version 4.0.2, cette opèration s'applique aussi aux noms de bases de donnèes.). Notez que si vous changez cette option, vous devrez repasser vos anciens noms de tables en minuscules avant de dèmarrer `mysqld`.

Si vous dèplacez les fichiers `MyISAM` d'un Windows á une architecture *nix, vous aurez peut être besoin d'utiliser l'utilitaire '`mysql_fix_extensions`' pour corriger la casse des extensions des fichiers dans chaque rèpertoire spècifique á la base de donnèes (minuscules '`.frm`', majuscules '`.MYI`' et '`.MYD`'). '`mysql_fix_extensions`' peut être trouvè dans le sous-rèpertoire '`script`'.

6.1.4 Variables utilisateur

MySQL supporte les variables utilisateur spècifiques á la connexion avec la syntaxe `@variablename`. Un nom de variable consiste de caractéres alpha-numèriques, basès sur le jeu de caractéres courant, de '_', '$', et '.'. Le jeu de caractéres par dèfaut est ISO-8859-1 Latin1. Cette valeur peut être changèe en utilisant l'option `--default-character-set` de `mysqld`. Voir Section 4.6.1 [Character sets], page 286.

Les variables n'ont pas besoin d'être initialisès. Elles sont á `NULL` par dèfaut et peuvent contenir un entier, un rèel ou une chaîne. Toutes les variables d'un thread sont automatiquement libèrèes lorsque le thread se termine.

Vous pouvez dèclarer une variable avec la syntaxe de `SET` :

```
SET @variable= { expression entier | expression rèel | expression chaîne }
[,@variable= ...].
```

Vous pouvez aussi assigner une valeur á une variable avec d'autres commande que `SET`. Par contre, dans ce cas lá, l'opèrateur d'assignation est `:=` au lieu de `=`, parce que `=` est rèservè aux comparaisons dans les requêtes autres que `SET` :

```
mysql> SELECT @t1:=(@t2:=1)+@t3:=4,@t1,@t2,@t3;
+----------------------+------+------+------+
| @t1:=(@t2:=1)+@t3:=4 | @t1  | @t2  | @t3  |
+----------------------+------+------+------+
|                    5 |    5 |    1 |    4 |
+----------------------+------+------+------+
```

Les variables utilisateur peuvent être utilisès lá oú les expressions sont allouèes. Notez que cela n'inclut pas pour l'instant les contextes oú un nombre est explicitement requis, comme ce qui est le cas avec la clause `LIMIT` dans une requête `SELECT`, ou la clause `IGNORE nombre LINES` dans une requête `LOAD DATA`.

Note : dans une requête `SELECT`, chaque expression est n'èvaluèe que lors de l'envoi au client. Cela signifie que pour les clauses `HAVING`, `GROUP BY`, ou `ORDER BY`, vous ne pouvez vous rèfèrer á une expression qui implique des variables qui sont dèfinies dans la partie `SELECT`. Par exemple, la requête suivante ne produira pas le rèsultat escomptè :

```
mysql> SELECT (@aa:=id) AS a, (@aa+3) AS b FROM nom_de_table HAVING b=5;
```

La raison est que `@aa` ne contiendra pas la valeur de la ligne courante, mais celle de `id` pour la derniére ligne acceptèe.

6.1.5 Variables systéme

A partir de la version 4.0.3 de MySQL, nous fournissons un meilleur accés á beaucoup de variables systéme et variables de connexion. On peut changer la plupart d'entre elle sans avoir á stopper le serveur.

Il y'a deux sortes de variables systéme : Les variables spècifiques aux threads (ou á la connexion) qui sont uniques á la connexion courante et les variables globales qui sont utilisèes pour configurer les èvénements globaux. Les variables globales sont aussi utilisèes pour configurer les valeurs initales des variables spècifiques au threads pour les nouvelles connexions.

Lorsque `mysqld` dèmarre, toutes les variables globales sont initialisèes á partir des arguements passès en ligne de commande et des fichiers de configuration. Vous pouvez changer ces valeurs avec la commande `SET GLOBAL`. Lorsqu'un nouveau thread est crèè, les variables spècifiques aux threads sont initialisèes á partir des variables globales et ne changeront pas même si vous utilisez la commande `SET GLOBAL`.

Pour dèfinir la valeur d'une variable `GLOBAL`, vous devez utiliser l'une des syntaxes suivantes : (Ici nous utilisons la variable `sort_buffer_size` á titre d'exemple)

```
SET GLOBAL sort_buffer_size=valeur;
SET @@global.sort_buffer_size=valeur;
```

Pour dèfinir la valeur d'une variable `SESSION`, vous devez utiliser l'une des syntaxes suivantes :

```
SET SESSION sort_buffer_size=valeur;
SET @@session.sort_buffer_size=valeur;
SET sort_buffer_size=valeur;
```

Si vous ne spècifiez pas `GLOBAL` ou `SESSION` alors `SESSION` est utilisè. Voir Section 5.5.6 [SET OPTION], page 396.

`LOCAL` est un synonyme de `SESSION`.

Pour rècupèrer la valeur d'une variable de type `GLOBAL` vous pouvez utiliser l'une des commandes suivantes :

```
SELECT @@global.sort_buffer_size;
SHOW GLOBAL VARIABLES like 'sort_buffer_size';
```

Pour rècupèrer la valeur d'une variable de type `SESSION` vous pouvez utiliser l'une des commandes suivantes :

```
SELECT @@session.sort_buffer_size;
SHOW SESSION VARIABLES like 'sort_buffer_size';
```

Lorsque vous **rècuperez** une valeur de variable avec la syntaxe `@@nom_variable` et que vous ne spècifiez pas `GLOBAL` ou `SESSION`, MySQL retournera la valeur spècifique au thread (`SESSION`) si elle existe. Sinon, MySQL retournera la valeur globale.

La raison d'imposer la prèsence du mot `GLOBAL` pour configurer une variable de type `GLOBAL` mais non pour la lire est pour être sur que vous n'aurez pas de problémes plus tard si vous voulez introduire ou effacer une variable spècifique au thread qui aurait le même nom. Dans ce cas, vous pourriez changer accidentellement l'ètat du serveur pour toutes les connexions (et non la votre uniquement).

Voilá la liste compléte de toutes les variables que que vous pouvez rècuperer ou modifier et une indication quant á l'utilisation de `GLOBAL` ou `SESSION` avec elles.

Nom de la variable	**Type de valeur**	**Type**
autocommit	bool	SESSION
big_tables	bool	SESSION
binlog_cache_size	num	GLOBAL
bulk_insert_buffer_size	num	GLOBAL SESSION
concurrent_insert	bool	GLOBAL
connect_timeout	num	GLOBAL
convert_character_set	string	SESSION

delay_key_write	OFF | ON | ALL	GLOBAL
delayed_insert_limit	num	GLOBAL
delayed_insert_timeout	num	GLOBAL
delayed_queue_size	num	GLOBAL
error_count	num	LOCAL
flush	bool	GLOBAL
flush_time	num	GLOBAL
foreign_key_checks	bool	SESSION
identity	num	SESSION
insert_id	bool	SESSION
interactive_timeout	num	GLOBAL | SESSION
join_buffer_size	num	GLOBAL | SESSION
key_buffer_size	num	GLOBAL
last_insert_id	bool	SESSION
local_infile	bool	GLOBAL
log_warnings	bool	GLOBAL
long_query_time	num	GLOBAL | SESSION
low_priority_updates	bool	GLOBAL | SESSION
max_allowed_packet	num	GLOBAL | SESSION
max_binlog_cache_size	num	GLOBAL
max_binlog_size	num	GLOBAL
max_connect_errors	num	GLOBAL
max_connections	num	GLOBAL
max_error_count	num	GLOBAL | SESSION
max_delayed_threads	num	GLOBAL
max_heap_table_size	num	GLOBAL | SESSION
max_join_size	num	GLOBAL | SESSION
max_sort_length	num	GLOBAL | SESSION
max_tmp_tables	num	GLOBAL
max_user_connections	num	GLOBAL
max_write_lock_count	num	GLOBAL
myisam_max_extra_sort_file_size	num	GLOBAL | SESSION
myisam_max_sort_file_size	num	GLOBAL | SESSION
myisam_sort_buffer_size	num	GLOBAL | SESSION
net_buffer_length	num	GLOBAL | SESSION
net_read_timeout	num	GLOBAL | SESSION
net_retry_count	num	GLOBAL | SESSION

net_write_timeout	num	GLOBAL SESSION	
query_cache_limit	num	GLOBAL	
query_cache_size	num	GLOBAL	
query_cache_type	enum	GLOBAL	
read_buffer_size	num	GLOBAL SESSION	
read_rnd_buffer_size	num	GLOBAL SESSION	
rpl_recovery_rank	num	GLOBAL	
safe_show_database	bool	GLOBAL	
server_id	num	GLOBAL	
slave_compressed_protocol	bool	GLOBAL	
slave_net_timeout	num	GLOBAL	
slow_launch_time	num	GLOBAL	
sort_buffer_size	num	GLOBAL SESSION	
sql_auto_is_null	bool	SESSION	
sql_big_selects	bool	SESSION	
sql_big_tables	bool	SESSION	
sql_buffer_result	bool	SESSION	
sql_log_binlog	bool	SESSION	
sql_log_off	bool	SESSION	
sql_log_update	bool	SESSION	
sql_low_priority_updates	bool	GLOBAL SESSION	
sql_max_join_size	num	GLOBAL SESSION	
sql_quote_show_create	bool	SESSION	
sql_safe_updates	bool	SESSION	
sql_select_limit	bool	SESSION	
sql_slave_skip_counter	num	GLOBAL	
sql_warnings	bool	SESSION	
table_cache	num	GLOBAL	
table_type	enum	GLOBAL SESSION	
thread_cache_size	num	GLOBAL	
timestamp	bool	SESSION	
tmp_table_size	enum	GLOBAL SESSION	
tx_isolation	enum	GLOBAL SESSION	
version	string	GLOBAL	
wait_timeout	num	GLOBAL SESSION	
warning_count	num	LOCAL	
unique_checks	bool	SESSION	

Les variables suivies d'un `num` peuvent se voir affecter une valeur numèrique. Celles marquèes avec `bool` peuvent prendre 0, 1, `ON` ou `OFF`. Les variables qui sont du type `enum` doivent nor-

malement comporter au moins l'une des valeurs disponibles pour la variable, mais peuvent aussi contenir les index des membres de la collection. (Le premier index est 0).

Voilá une description de quelques variables :

Variable	Description
identity	Alias pour last_insert_id (compatibilitè Sybase)
sql_low_priority_updates	Alias pour low_priority_updates
sql_max_join_size	Alias pour max_join_size
delay_key_write_for_all_tables	Si cette variable et delay_key_write sont dèfinies, alors toutes les nouvelles tables MyISAM qui sont ouvertes utiliseront les ècritures de clefs reportèes.
version	Alias for VERSION() (compatibilitè Sybase (?))

Une dèscription des autres variables peut être trouvèe dans la section des options de dèmarrage, la description de `SHOW VARIABLES` et dans la section de `SET`. Voir Section 4.1.1 [Command-line options], page 192. Voir Section 4.5.6.4 [SHOW VARIABLES], page 273. Voir Section 5.5.6 [SET OPTION], page 396.

6.1.6 Syntaxe des commentaires

Le serveur MySQL supporte les commentaires de la forme `# jusqu'á la fin de la ligne`, `-- jusqu'á la fin de la ligne` et `/* dans la ligne ou sur plusieurs lignes */` :

```
mysql> SELECT 1+1;     # Ce commentaire se continue jusqu'á la fin de la ligne
mysql> SELECT 1+1;     -- Ce commentaire se continue jusqu'á la fin de la ligne
mysql> SELECT 1 /* Ceci est un commentaire dans la ligne */ + 1;
mysql> SELECT 1+
/*
Ceci est un commentaire
sur plusieurs lignes
*/
1;
```

Notez que le style de commentaire `--` requiert au moins un espace aprés le second tiret !

Même si le serveur interpréte correctement les syntaxes de commentaires dècrites plus haut, il y'a quelques limitations dans la façon oú le client interpréte les commentaires multi-lignes :

- Les guillemets (simples et doubles) sont considèrès comme des indications de dèbut de chaîne, même dans un commentaire. Si le guillemet n'est pas refermè (par un second guillemet), l'analyseur ne rèalisera pas que le commentaire est fini. Si vous utilisez mysql interactivement, vous pouvez vous en apercevoir, car il va modifier l'invite de commande de `mysql>` en `'>` ou `">`.
- Un point-virgule sert á indiquer la fin de la commande SQL, et tout ce qui suit un point-virgule est considèrè comme ètant une nouvelle requête.

Ces limitations s'appliquent aussi bien á `mysql` en ligne de commande, que lorsque vous demandez á `mysql` de lire des commandes depuis un fichier (`mysql < un-fichier`).

MySQL ne supporte les commentaires du style '`--`' ANSI SQL que si le second tiret est suivi d'un espace. Voir Section 1.7.4.7 [ANSI diff comments], page 42.

6.1.7 Est-ce que MySQL est sensible aux mots rèservès ?

Un probléme rècurrent provient de la tentative de crèation de tables avec des noms de colonnes qui sont des types de champs ou des fonctions natives de MySQL, comme `TIMESTAMP` ou `GROUP`. Il vous st permis de le faire (par exemple `ABS` est permis comme nom de colonne), mais les espaces ne sont pas permis entre le nom d'une fonction et la premiére '`(`' suivante lors de l'utilisation de fonctions qui sont aussi des noms de colonnes.

Les mots suivants sont explicitement rèservès en MySQL. La plupart sont interdits par ANSI SQL92 en tant que nom de colonnes ou de tables (par exemple, `GROUP`). Quelques uns sont rèservès parce que MySQL en a besoin et utilise (actuellement) un analyseur `yacc` :

Word	Word	Word
ADD	ALL	ALTER
ANALYZE	AND	AS
ASC	ASENSITIVE	AUTO_INCREMENT
BDB	BEFORE	BERKELEYDB
BETWEEN	BIGINT	BINARY
BLOB	BOTH	BTREE
BY	CALL	CASCADE
CASE	CHANGE	CHAR
CHARACTER	CHECK	COLLATE
COLUMN	COLUMNS	CONNECTION
CONSTRAINT	CREATE	CROSS
CURRENT_DATE	CURRENT_TIME	CURRENT_TIMESTAMP
CURSOR	DATABASE	DATABASES
DAY_HOUR	DAY_MINUTE	DAY_SECOND
DEC	DECIMAL	DECLARE
DEFAULT	DELAYED	DELETE
DESC	DESCRIBE	DISTINCT
DISTINCTROW	DIV	DOUBLE
DROP	ELSE	ELSEIF
ENCLOSED	ERRORS	ESCAPED
EXISTS	EXPLAIN	FALSE
FIELDS	FLOAT	FOR
FORCE	FOREIGN	FROM
FULLTEXT	GRANT	GROUP
HASH	HAVING	HIGH_PRIORITY
HOUR_MINUTE	HOUR_SECOND	IF
IGNORE	IN	INDEX
INFILE	INNER	INNODB
INOUT	INSENSITIVE	INSERT
INT	INTEGER	INTERVAL
INTO	IS	ITERATE
JOIN	KEY	KEYS
KILL	LEADING	LEAVE
LEFT	LIKE	LIMIT

LINES	LOAD	LOCALTIME
LOCALTIMESTAMP	LOCK	LONG
LONGBLOB	LONGTEXT	LOOP
LOW_PRIORITY	MASTER_SERVER_ID	MATCH
MEDIUMBLOB	MEDIUMINT	MEDIUMTEXT
MIDDLEINT	MINUTE_SECOND	MOD
MRG_MYISAM	NATURAL	NOT
NULL	NUMERIC	ON
OPTIMIZE	OPTION	OPTIONALLY
OR	ORDER	OUT
OUTER	OUTFILE	PRECISION
PRIMARY	PRIVILEGES	PROCEDURE
PURGE	READ	REAL
REFERENCES	REGEXP	RENAME
REPEAT	REPLACE	REQUIRE
RESTRICT	RETURN	RETURNS
REVOKE	RIGHT	RLIKE
RTREE	SELECT	SENSITIVE
SEPARATOR	SET	SHOW
SMALLINT	SOME	SONAME
SPATIAL	SPECIFIC	SQL_BIG_RESULT
SQL_CALC_FOUND_ROWS	SQL_SMALL_RESULT	SSL
STARTING	STRAIGHT_JOIN	STRIPED
TABLE	TABLES	TERMINATED
THEN	TINYBLOB	TINYINT
TINYTEXT	TO	TRAILING
TRUE	TYPES	UNION
UNIQUE	UNLOCK	UNSIGNED
UNTIL	UPDATE	USAGE
USE	USER_RESOURCES	USING
VALUES	VARBINARY	VARCHAR
VARCHARACTER	VARYING	WARNINGS
WHEN	WHERE	WHILE
WITH	WRITE	XOR
YEAR_MONTH	ZEROFILL	

Les symboles suivants (issus de la table ci-dessus) sont interdits par ANSI SQL mais permis par MySQL en tant que noms de colonnes ou de tables. Cela est du au fait que ces noms sont trés courants, et de nombreux programmeur les ont dèjá utilisès.

- ACTION
- BIT
- DATE
- ENUM
- NO
- TEXT
- TIME

- TIMESTAMP

6.2 Types de colonnes

MySQL supporte un grand nombre de types de colonnes, qui peuvent être rassemblès en trois groupes : les nombres, les dates et les chaînes de caractéres. Cette section prèsente les types disponibles et leurs tailles de stockage, puis prèsente en dètail chaque type. L'introduction est volontairement bréve. Une section plus prècise est dèdiée á chaque type, qui prèsente tous les formats valides.

Les types de colonnes de MySQL sont listès ci-dessous. Les codes suivants sont utilisès dans les descriptions :

`M` Indique la taille maximale d'affichage. Le maximum lègal est 255.

`D` S'applique aux nombres á virgule flottante, et indique le nombre de dècimales qui suivent la virgule. Le nombre maximum est de 30, mais ne doit pas être plus grand que `M`-2.

Les crochets ('`[`' et '`]`') indiquent les spècifications optionnelles.

Notez que si vous spècifiez ZEROFILL pour une colonne, MySQL ajoutera automatiquement l'attribut UNSIGNED á la colonne.

Attention : vous devez garder á l'esprit que lors de la soustraction de deux entiers dont l'un est de type `UNSIGNED`, le rèsultat ne sera pas signè ! Voir Section 6.3.5 [Cast Functions], page 470.

`TINYINT[(M)] [UNSIGNED] [ZEROFILL]`
Un trés petit entier. S'il est signè, sa valeur varie entre `-128` et `127`, sinon elle varie de `0` á `255`.

`BIT`
`BOOL` Ce sont des synonymes de `TINYINT(1)`.

`SMALLINT[(M)] [UNSIGNED] [ZEROFILL]`
Un petit entier. S'il est signè, sa valeur varie entre `-32768` et `32767`, sinon elle varie de `0` á `65535`.

`MEDIUMINT[(M)] [UNSIGNED] [ZEROFILL]`
Un entier de taille moyenne. S'il est signè, sa valeur varie entre `-8388608` et `8388607`, sinon elle varie entre `0` et `16777215`.

`INT[(M)] [UNSIGNED] [ZEROFILL]`
Un entier. S'il est signè, sa valeur varie entre `-2147483648` et `2147483647`, sinon elle varie entre `0` et `4294967295`.

`INTEGER[(M)] [UNSIGNED] [ZEROFILL]`
C'est un synonyme de `INT`.

`BIGINT[(M)] [UNSIGNED] [ZEROFILL]`
Un grand entier. S'il est signè, sa valeur varie entre `-9223372036854775808` et `9223372036854775807`, sinon elle varie entre `0` et `18446744073709551615`.

Quelques choses á savoir pour bien utiliser les colonnes de type `BIGINT` :

- Toute l'arithmètique est effectuèe en utilisant des `BIGINT` signès ou des valeurs de type `DOUBLE`, vous ne devez donc pas utiliser de grands entiers non-signès plus grand que `9223372036854775807` (63 bits) exceptè avec les fonctions sur les bits ! Si vous le faites, certains des derniers chiffres du rèsultat risquent d'être faux á cause des erreurs d'arrondissage lors de la conversion de `BIGINT` á `DOUBLE`.

  MySQL 4.0 peut gèrer les `BIGINT` dans les cas suivants :

  - Utilisation d'entiers pour stocker une grande valeur non-signèe dans une colonne `BIGINT`.
  - Avec `MIN(colonne_big_int)` and `MAX(colonne_big_int)`.
  - Lors de l'utilisation des opèrateurs (`+`, `-`, `*`, etc.) lorsque les deux opèrandes sont des entiers.
- Vous pouvez toujours stocker la valeur exacte d'un entier dans une colonne de type `BIGINT` en l'enregistrant en tant que chaîne. Dans ce cas, MySQL èffectuera une conversion chaîne á nombre qui ne fait entrer en jeu aucune reprèsentation intermèdiaire en rèel.
- '`-`', '`+`', et '`*`' utiliseront l'arithmètique `BIGINT` quand les deux arguments sont des valeurs entiéres ! Cela signifie que si vous multipliez deux grands entiers (ou des rèsultats de fonctions qui retournent des entiers) vous obtiendrez peut être des rèsultats inattendus quand le rèsultat est supèrieur á `9223372036854775807`.

`FLOAT(prècision) [UNSIGNED] [ZEROFILL]`

Un nombre á virgule flottante. `prècision`<=24 pour un nombre á virgule flottante de prècision simple, entre 25 et 53 pour une prècision double. Ces types correspondent aux types `FLOAT` et `DOUBLE` dècrits ci-dessus. FLOAT(X) a le même intervalle de validitè que `FLOAT` et `DOUBLE`, mais la taille d'affichage et le nombre de dècimale sont indèfinies.

Dans la version 3.23 de MySQL, c'est un vèritable nombre á virgule flottante. Dans les anciennes versions de MySQL, `FLOAT(prècision)` avait toujours deux dècimales.

Notez que l'utilisation du type `FLOAT` peut vous crèèr des problémes inattendus car tous les calculs internes de MySQL sont fait en double prècision. Voir Section A.5.6 [No matching rows], page 706.

Cette syntaxe est fournie pour assurer la compatibilitè avec ODBC.

`FLOAT[(M,D)] [UNSIGNED] [ZEROFILL]`

Un petit nombre á virgule flottante (prècision simple). L'intervalle de validitè va de `-3.402823466E+38` á `-1.175494351E-38`, 0 et de `1.175494351E-38` á `3.402823466E+38`. Si `UNSIGNED` est spècifiè, les valeurs nègatives sont interdites. `M` reprèsente la taille d'affichage et `D` est le nombre de dècimales. `FLOAT` sans arguments, ou avec un argument <= 24 reprèsente un nombre á virgule flottante de prècision simple.

`DOUBLE[(M,D)] [UNSIGNED] [ZEROFILL]`

Un nombre á virgule flottante (prècision double). L'intervalle de validitè va de `-1.7976931348623157E+308` á `-2.2250738585072014E-308`, 0 et de

2.2250738585072014E-308 á 1.7976931348623157E+308. Si UNSIGNED est spècifiè, les valeurs nègatives sont interdites. M reprèsente la taille d'affichage et D est le nombre de dècimales. FLOAT sans arguments, ou avec un argument compris entre 25 et 53 (inclus) reprèsente un nombre á virgule flottante de prècision double.

DOUBLE PRECISION[(M,D)] [UNSIGNED] [ZEROFILL]
REAL[(M,D)] [UNSIGNED] [ZEROFILL]
C'est un synonyme de DOUBLE.

DECIMAL[(M[,D])] [UNSIGNED] [ZEROFILL]
Un nombre á virgule flottante. Il doit être signè. Ce type se comporte comme une colonne de type CHAR : la valeur est stockèe comme une chaîne, chaque caractére reprèsentant un chiffre de la valeur. La virgule et le signe '-' des nombres nègatifs ne sont pas comptès dans l'option M (mais de l'espace de stockage est reservè pour eux). Si D vaut 0, les valeurs n'auront pas de valeur dècimale. L'intervalle de validitè des valeurs DECIMAL est le même que DOUBLE, mais il peut être limitè par les valeurs choisies pour M et D. Si UNSIGNED est spècifiè, les valeurs nègatives sont interdites.

Si D est ommis, la valeur par dèfaut est 0. Si M est ommis, la valeur par dèfaut est 10.

Avant la version 3.23 de MySQL, l'argument M doit inclure l'espace requis pour le signe et le point des dècimales.

DEC[(M[,D])] [UNSIGNED] [ZEROFILL]
NUMERIC[(M[,D])] [UNSIGNED] [ZEROFILL]
Ce sont des synonymes de DECIMAL.

DATE

Une date. L'intervalle de validitè va de '1000-01-01' á '9999-12-31'. MySQL affiche les valeurs DATE au format 'AAAA-MM-JJ' (annèe-mois-jour), mais vous pouvez assigner des valeurs aux colonnes DATE en utilisant diffèrents formats numèriques ou de chaînes de caractéres.

DATETIME

Une combinaison de date et d'heure. L'intervalle de validitè va de '1000-01-01 00:00:00' á '9999-12-31 23:59:59'. MySQL affiche les valeurs DATETIME au format 'AAAA-MM-JJ HH:MM:SS' (annèe-mois-jour heure:minutes:secondes), mais vous pouvez assigner des valeurs aux colonnes DATETIME en utilisant diffèrents formats numèriques ou de chaînes de caractéres. Voir Section 6.2.2.2 [DATETIME], page 424.

TIMESTAMP[(M)]
Un timestamp. L'intervalle de validitè va de '1970-01-01 00:00:00' á quelque part durant l'annèe 2037. MySQL affiche les valeurs de type TIMESTAMP au format AAAAMMJJHHMMSS, AAMMJJHHMMSS, AAAAMMJJ, ou AAMMJJ, suivant que le paramétre M vaut 14 (ou omis), 12, 8, ou 6, mais vous pouvez assigner des valeurs aux colonnes TIMESTAMP sous forme de nombre ou de chaînes. Une colonne de type TIMESTAMP est pratique pour enregistrer une date, lors d'une

commande `INSERT` ou `UPDATE`, car elle est automatiquement mise á la date et l'heure du moment de la commande, si vous ne fournissez pas de valeur vous-même. Vous pouvez aussi lui donner la date et l'heure courante en lui assignant la valeur `NULL`. Voir Section 6.2.2 [Date and time types], page 423.

Un `TIMESTAMP` est toujours stockè sur 4 octets. L'argument `M` n'affecte que le mode d'affichage de ce type de colonne.

Notez bien que les colonnes de type `TIMESTAMP(X)` oú `X` vaut `8` ou `14` sont considèrèes comme des nombres tandis que les autres colonnes `TIMESTAMP(X)` sont considèrèes comme des chaînes. Ceci est fait pour s'assurer que l'on peut exporter et importer les tables avec ces types. Voir Section 6.2.2.2 [`DATETIME`], page 424.

`TIME`

Une heure. L'intervalle de validitè va de `'-838:59:59'` á `'838:59:59'`. MySQL affiche les colonnes de type `TIME` au format `'HH:MM:SS'`, mais vous pouvez assigner une valeur de type `TIME` en lui passant des chaînes ou des entiers. Voir Section 6.2.2.3 [`TIME`], page 428.

`YEAR[(2|4)]`

Une annèe au format 2 ou 4 chiffres (par dèfaut, c'est 4 chiffres). L'intervalle de validitè va de `1901` á `2155`, `0000` pour le format á 4 chiffres, et de `1970` á `2069` pour le format á deux chiffres. MySQL affiche les valeurs de type `YEAR` au format `AAAA`, mais vous pouvez leur assigner des chaînes ou des nombres. (le type `YEAR` est nouveau depuis la version 3.22 de MySQL) Voir Section 6.2.2.4 [`YEAR`], page 429.

`[NATIONAL] CHAR(M) [BINARY]`

Une chaîne de caractéres de taille fixe, qui est toujours complètèe á droite, par des espaces, lors du stockage. Le paramétre `M` peut valoir de 0 á 255 caractéres. (1 á 255 dans les versions antèrieures á la 3.23) Les espaces terminaux sont supprimès lorsque la chaîne est lue dans la base. Les valeurs de type `CHAR` sont triées et comparées sans tenir compte de la casse et en utilisant le jeu de caractéres par dèfaut. Toutefois, vous pouvez utiliser l'opèrateur BINARY pour effectuer des recherches sensibles á la casse.

`NATIONAL CHAR` (forme courte : `NCHAR`) est la dènomination ANSI SQL pour les colonnes de type `CHAR` qui doivent utiliser le jeu de caractéres par dèfaut. C'est la configuration par dèfaut de MySQL.

MySQL autorise la crèation de colonne de type `CHAR(0)`. Cela ne sert rèellement que si vous devez être compatible avec une vieille application, dont le bon fonctionnement repose sur l'existence de la colonne, mais qui n'utilise pas vraiment ses valeurs. C'est aussi pratique lorsque vous devez stocker un boolèen (ou une valeur á deux ètats) dans une colonne : une colonne de type `CHAR(0)`, qui n'a pas l'attribut NOT NULL, n'occupera qu'un seul octet, et peut prendre deux valeurs : `NULL` et `""`. Voir Section 6.2.3.1 [`CHAR`], page 430.

`CHAR` est une forme courte de `CHARACTER`.

`CHAR` C'est un synonyme de `CHAR(1)`.

`[NATIONAL] VARCHAR(M) [BINARY]`

Une chaîne de caractéres de longueur variable. **NOTE :** les espaces terminaux sont supprimès lors du stockage des valeurs (ce qui différe des spècifications ANSI SQL). L'intervalle de taille de M va de 1 á 255 caractéres. Les valeurs de type `VARCHAR` sont triées et comparèes sans tenir compte de la casse et en utilisant le jeu de caractéres par dèfaut. Toutefois, vous pouvez utiliser l'opèrateur `BINARY` pour effectuer des recherches sensibles á la casse. Voir Section 6.5.3.1 [Silent column changes], page 511.

`VARCHAR` est un forme courte de `CHARACTER VARYING`. Voir Section 6.2.3.1 [CHAR], page 430.

`TINYBLOB`
`TINYTEXT`

Une colonne `BLOB` ou `TEXT` avec une longueur maximale de 255 (2^8 - 1) caractéres. Voir Section 6.5.3.1 [Silent column changes], page 511. Voir Section 6.2.3.2 [BLOB], page 430.

`BLOB`
`TEXT`

Une colonne `BLOB` ou `TEXT` avec une longueur maximale de 65535 (2^16 - 1) caractéres. Voir Section 6.5.3.1 [Silent column changes], page 511. Voir Section 6.2.3.2 [BLOB], page 430.

`MEDIUMBLOB`
`MEDIUMTEXT`

Une colonne `BLOB` ou `TEXT` avec une longueur maximale de 16777215 (2^24 - 1) caractéres. Voir Section 6.5.3.1 [Silent column changes], page 511. Voir Section 6.2.3.2 [BLOB], page 430.

`LONGBLOB`
`LONGTEXT`

Une colonne `BLOB` ou `TEXT` avec une longueur maximale de 4294967295 (2^32 - 1) caractéres. Voir Section 6.5.3.1 [Silent column changes], page 511. Notez que puisque le protocole serveur/client et les tables de type MyISAM ont une limitation de 16M par paquet communiquè / ligne, vous ne pourrez utiliser la totalitè de la longueur de ce type. Voir Section 6.2.3.2 [BLOB], page 430.

`ENUM('valeur1','valeur2',...)`

Une ènumèration. Une chaîne de caractéres qui ne peut prendre qu'une valeur, issue d'une liste de valeurs `'valeur1'`, `'valeur2'`, `...`, `NULL` ou la valeur spèciale d'erreur `""`. Un `ENUM` peut avoir un maximum de 65535 valeurs distinctes. Voir Section 6.2.3.3 [ENUM], page 432.

`SET('valeur1','valeur2',...)`

Un ensemble. Une chaîne de caractéres qui a zèro ou plusieurs valeurs issues d'une liste : `'valeur1'`, `'valeur2'`, `...` Un `SET` peut avoir au maximum 60 èlèments. Voir Section 6.2.3.4 [SET], page 433.

6.2.1 Types numèriques

MySQL supporte tous les types numèriques de la norme ANSI/ISO SQL92. Ceux ci reprèsentent les types numèriques exacts (`NUMERIC`, `DECIMAL`, `INTEGER`, et `SMALLINT`), ainsi que les types approchès (`FLOAT`, `REAL`, et `DOUBLE PRECISION`). Le mot clef `INT` est un synonyme de `INTEGER`, et le mot clef `DEC` est un synonyme de `DECIMAL`.

Les types `NUMERIC` et `DECIMAL` sont considèrès comme identiques par MySQL, comme l'autorise le standard SQL92. Ils sont utilisèes par des valeurs dont il est primordial de conserver la prècision exacte, comme pour des donnèes financiéres. Lorsque vous dèclarez des colonnes avec l'un de ces types, vous pouvez indiquer la prècision et l'èchelle comme ceci :

```
salaire DECIMAL(5,2)
```

Dans cet exemple, `5` (`prècision`) reprèsente le nombre de dècimales signifiantes qui seront stockèes pour les valeurs, et `2` (`èchelle`) reprèsente le nombre de chiffres qui seront stockès aprés le point des dècimales. Dans ce cas, toutefois, l'intervalle des valeurs pouvant être stockès dans la colonne `salaire` varie de `-99.99` á `99.99`. (MySQL peut actuellement stocker des nombres allant jusqu'á `999.99` dans cette colonne car il n'a pas besoin de l'espace qui sert á stocker le signe pour un nombre positif)

En ANSI/ISO SQL92, la syntaxe `DECIMAL(p)` est èquivalente á `DECIMAL(p,0)`. De maniére similaire, la syntaxe `DECIMAL` est èquivalente á to `DECIMAL(p,0)`, oú l'implèmentation est autorisèe á choisir la valeur de `p`. MySQL ne supporte pas actuellement ces formes variantes des types de donnèes `DECIMAL/NUMERIC`. Cela ne pose pas gènèralement de grand problémes, vu que le principal bènèfice de ces types est de pouvoir contrôler explicitement la prècision et l'èchelle.

Les valeurs de type `DECIMAL` et `NUMERIC` sont stockèes sous forme de chaînes de caractéres, plutôt que comme des nombres á virgule flottante, afin de prèserver la prècision dècimale des valeurs. Un caractére est donc nècessaire pour chaque chiffre, plus la virgule (si `scale > 0`), et le signe moins '`-`' (pour les nombres nègatifs). Si `scale` vaut 0, les valeurs de type `DECIMAL` et `NUMERIC` ne comporteront pas de valeur dècimale, ni de virgule.

L'intervalle de validitè maximale des valeurs de type `DECIMAL` et `NUMERIC` est le même que pour le type `DOUBLE`, mais l'intervalle rèel peut être limitè par le choix des paramétres `prècision` et `scale`. Lorsqu'une valeur ayant trop de dècimales est affectèe á une colonne, la valeur est arrondie á `scale` dècimales. Lorsqu'une valeur est hors des limites de validitè de la colonne `DECIMAL` ou `NUMERIC`, MySQL enregistre la plus grande valeur qu'il peut á la place.

En extension de la norme ANSI/ISO SQL92, MySQL supporte aussi les types entiers `TINYINT`, `MEDIUMINT`, et `BIGINT`, comme prèsentè ci-dessus. Un autre extension supportèe par MySQL permet de spècifier optionnellement la taille d'affichage, sous la forme d'une valeur entiére entre parenthéses, juste aprés le mot clè spècifiant le type (par exemple, INT(4)). Cette spècification de taille est utilisèe pour remplir á gauche, avec le caractére de remplissage par dèfaut, les nombres dont la taille est infèrieure á celle spècifièe mais uniquement á l'affichage : cela ne rèduit pas l'intervalle de validitè des valeurs qui peuvent être stockèes dans la colonne.

Lorsqu'elle est utilisèe avec l'attribut de colonne optionnel `ZEROFILL`, le caractére de remplissage par dèfaut est remplacè par le caractére zèro. Par exemple, pour une colonne dont le type est `INT(5) ZEROFILL`, la valeur `4` sera lue `00004`.

Notez qye si vous stockez des nombres plus grands que la taille maximale d'affichage, vous pouvez rencontrer des problémes lors de jointures de tables particuliérement compliquèes, surtout si MySQL gènère des tables temporaires : dans ce cas, MySQL pense que les donnèes ètaient limitèes par l'affichage.

Tous les types entiers ont un attribut optionnel (non-standard) `UNSIGNED` (non-signè, en français). Les valeurs non-signèes peuvent être utilisèes pour n'autoriser que des valeurs positives dans une colonne, ou bien pour exploiter un intervalle de validitè plus haut. Depuis la version 4.0.2 de MySQL, les nombres á virgule flottante peuvent aussi être `UNSIGNED` Comme avec les types entiers, cet attribut interdit les valeurs nègatives dans la colonne, mais n'èléve pas l'intervalle de validitè.

Le type FLOAT est utilisè pour reprèsenter des donnèes numèriques approchèes. La norme ANSI/ISO SQL92 permet la spècification optionnelle de la prècision (mais pas de l'intervalle de validitè) en fournissant le nombre de dècimales voulues aprés la spècification de type, et entre parenthéses. L'implèmentation de MySQL supporte aussi le paramètrage de la prècision. Si le mot clè `FLOAT` est utilisè pour une colonne sans prècision supplèmentaire, MySQL utilise quatre octets pour stocker les valeurs. Une syntaxe alternative existe aussi, elle utilise deux paramétre optionnel aprés le mot clè `FLOAT`. Avec cette option, le premier nombre reprèsente toujours la taille de stockage nècessaire pour la valeur, et le second nombre reprèsente le nombre de chiffres á stocker et afficher, aprés la virgule dècimale (comme pour les types `DECIMAL` et `NUMERIC`). Lorsque MySQL stocke un nombre pour une telle colonne, et que cette valeur a plus de dècimale que requis, la valeur est arrondie pour èliminer les chiffres surnumèraires.

Les types `REAL` et `DOUBLE PRECISION` n'acceptent pas de paramètrage de la prècision. En extension du standard ANSI/ISO SQL92, MySQL reconnaît `DOUBLE` comme un synonymee du type `DOUBLE PRECISION`. Contrairement á la norme qui requiert que `REAL` soit plus petit que `DOUBLE PRECISION`, MySQL implèmente ces deux types comme des nombres á virgule flottante de 8 octets, en double prècision (lorsque le mode "ANSI" n'est pas activè). Pour une portabilitè maximale, les applications rèclamant le stockage de nombres approchè doivent utiliser les types `FLOAT` ou `DOUBLE PRECISION` sans spècification de prècision ou de nombre de dècimales.

Lorsque MySQL doit stocker une valeur qui est hors de l'intervalle de validitè d'une colonne, il raméne la valeur á la plus proche possible, et stocke cette valeur. Par exemple, l'intervalle de validitè d'une colonne d'entiers `INT` va de `-2147483648` á `2147483647`. Si vous essayez d'insèrer `-9999999999` dans une colonne de ce type, la valeur sera ramenèe á la plus proche possible, c'est á dire `-2147483648`. De même, si vous essayez d'insèrer `9999999999`, `2147483647` sera stockè á la place.

Si la colonne `INT` posséde l'attribut `UNSIGNED`, l'intervalle de validitè est aussi large, mais les valeurs extrêmes se dècalent vers `0` et `4294967295`. Si vous essayez de stocker `-9999999999` et `9999999999` dans cette colonne, vous obtiendrez respectivement `0` et `4294967296`.

Les dèpassements de capacitè entraînant des tronquage sont affichès comme des alertes ("warnings") lors de l'utilisation des commandes `ALTER TABLE`, `LOAD DATA INFILE`, `UPDATE`, et les insertions `INSERT` multiples.

Type	Octets	De	A
TINYINT	1	-128	127
SMALLINT	2	-32768	32767
MEDIUMINT	3	-8388608	8388607
INT	4	-2147483648	2147483647
BIGINT	8	-9223372036854775808	9223372036854775807

6.2.2 Les types date et heure

Les types dates et heures sont DATETIME, DATE, TIMESTAMP, TIME, et YEAR. Chacun d'eux á une èchelle de valeurs lègales, de même que la valeur "zèro" quand vous spècifiez une valeur illègale. A noter que MySQL vous permet d'enregistrer certaines dates qui ne sont pas strictement lègales, par exemple 1999-11-31. La raison est que nous pensons que la vèrification des dates est á faire niveau application. Pour accèlèrer les tests, MySQL vèrifie juste que le mois est entre 0 et 12 et que le jour est entre 0 et 31. Les intervalles prècèdentes sont dèfinies de cette façon car MySQL vous permet d'enregistrer dans une colonne DATE ou DATETIME, des dates oú le jour de la semaine ou le jour du mois est zèro. C'est extrêmement utile pour les applications oú vous avez besoin d'enregistrer une date d'anniversaire pour laquelle vous n'avez pas la date exacte. Dans ce cas, vous enregistrez simplement la date comme 1999-00-00 ou 1999-01-00. (Vous ne devez pas vous attendre á obtenir de valeurs correctes de fonctions tel que DATE_SUB() ou DATE_ADD pour des dates comme cela.)

Voici quelques considèrations á garder á l'esprit quand vous manipulerez ce type de champs :

- MySQL extrait les valeurs d'un champ date ou heure dans un format standard, mais essaye d'interprèter une grande variètès de format pour les valeurs que vous donnez (par exemple, quand vous essayez de comparer ou attribuer une valeur á un champ de type date ou heure). Nèanmoins, seul les formats dècrits dans les sections suivantes sont disponibles. Il est attendu que vous fournissiez des valeurs lègales et des erreurs imprèvues peuvent survenir si vous utilisez d'autre formats.
- Même si MySQL essaye d'interpreter les valeurs sous diffèrents formats, il s'attend toujours á ce que l'annèe soit dans la partie gauche de la valeur. Les dates doivent êtres donnèes sous la forme annèe-mois-jour (exemple : 98-09-04), au lieu de mois-jour-annèe ou jour-mois-annèe qui sont trés utilisès ailleurs (comme 09-04-98 ou '04-09-98').
- MySQL convertit automatiquement une date ou heure en nombre si la valeur est utilisèe dans un contexte numèrique et vice versa.
- Lorsque MySQL rencontre une valeur hors d'intervalle pour un type date ou heure qui est donc illègale pour ce type (voir le dèbut de cette section), il la convertit á la valeur "zèro" de ce type. (L'exception est que les valeurs hors intervalles pour les champs TIME sont coupèes á la limite approprièe des valeurs de TIME.) Le tableau suivant prèsente le format de la valeur "zèro" de chaque type :

Column type	valeur du "zèro"
DATETIME	'0000-00-00 00:00:00'
DATE	'0000-00-00'
TIMESTAMP	00000000000000 (la longueur dèpend de la taille de l'affichage)

```
TIME          '00:00:00'
YEAR          0000
```

- La valeur "zèro" est spèciale, mais vous pouvez l'enregistrer ou vous y rèfèrer explicitement en utilisant les valeurs contenues dans le tableau ci dessus. Vous pouvez aussi le faire en utilisant la valeur `'0'` ou `0` qui est plus facile á manipuler.
- La date ou le temps "Zèro" utilisè avec `MyODBC` est automatiquement convertie en `NULL` á partir de la version 2.50.12 de `MyODBC`, car ODBC ne peut manipuler de telles valeurs.

6.2.2.1 An 2000 et les types date

MySQL lui même est compatible an 2000. (voir Section 1.2.5 [Year 2000 compliance], page 10), mais les valeurs manipulèes par MySQL peuvent ne pas l'être. N'importe quelle valeur n'ayant que deux chiffres pour reprèsenter l'annèe est ambigu, car le siécle n'est pas prècisè. Ces valeurs doivent être interprètèes comme des valeurs á 4 chiffres, car MySQL stocke les annèes en interne en utilisant 4 chiffres.

Pour les types `DATETIME`, `DATE`, `TIMESTAMP`, et `YEAR`, MySQL interpréte les dates ambigus en se basant sur les régles suivantes :

- Les valeurs d'annèes comprises dans l'intervalle `00-69` sont converties en `2000-2069`.
- Les valeurs d'annèes comprises dans l'intervalle `70-99` sont converties en `1970-1999`.

Gardez bien á l'esprit que ces régles ne sont que la meilleure approximation possible d'une valeur. Si l'heuristique proposèe par MySQL ne fournit pas les valeurs attendues, vous devrez fournir une valeur sans ambiguïtè. (á 4 chiffres)

`ORDER BY` ordonnera correctement les types `YEAR/DATE/DATETIME` á deux chiffres.

Notez aussi que quelques fonctions comme `MIN()` et `MAX()` convertiront un `TIMESTAMP/DATE` en nombre. Cela signifie qu'un timestamp avec une annèe á deux chiffres ne donneront pas de rèsultats corrects avec ces fonctions. Une solution dans ce cas est de convertir le `TIMESTAMP/DATE` en une annèe á 4 chiffres ou d'utiliser quelque chose comme `MIN(DATE_ADD(timestamp,INTERVAL 0 DAYS))`.

6.2.2.2 Les types `DATETIME`, `DATE`, et `TIMESTAMP`

Les types `DATETIME`, `DATE`, et `TIMESTAMP` sont liès. Cette section dècrit leurs caractèristiques, leur similaritès et leurs diffèrences.

Le type `DATETIME` est prèvu lorsque vous souhaitez stocker une date et une heure. MySQL affiche les valeurs de type `DATETIME` au format '`AAAA-MM-JJ HH:MM:SS`'. L'intervalle de validitè va de '`1000-01-01 00:00:00`' á '`9999-12-31 23:59:59`'. ("validitè" signifie que même si d'autres valeurs plus anciennes peuvent être manipulèes, il n'est pas garantit qu'elles le seront).

Le type `DATE` est prèvu lorsque vous souhaitez stocker une date. MySQL affiche les valeurs de type `DATE` au format '`AAAA-MM-JJ`'. L'intervalle de validitè va de `'1000-01-01'` á `'9999-12-31'`.

Le type `TIMESTAMP` est prèvu pour stocker automatiquement l'heure courante lors d'une commande `INSERT` ou `UPDATE`. Si vous avez plusieurs colonnes de type `TIMESTAMP`, seule la premiére colonne sera mise á jour automatiquement.

La modification automatique de la premiére colonne de type TIMESTAMP survient si l'une des conditions suivantes est remplie :

- La colonne n'est pas spècifièe explicitement dans la commande INSERT ou LOAD DATA INFILE.
- La colonne n'est pas spècifièe explicitement dans la commande UPDATE et d'autres colonnes changent de valeurs (Notez qu'une commande UPDATE qui affecte une valeur qui est dèjá celle de la colonne sera ignorèe, et la colonne TIMESTAMP ne sera pas modifièe, car la ligne n'est pas á proprement parlèe modifièe. MySQL ignore alors ces modifications pour des raisons d'efficacitè).
- La colonne TIMESTAMP est spècifièe explicitement á NULL.

Les autres colonnes de type TIMESTAMP, hormis la premiére, peuvent aussi prendre la valeur courante. Affectez lui alors la valeur NULL ou la fonction NOW().

Vous pouvez affecter á n'importe quelle colonne de type TIMESTAMP une valeur diffèrente de l'heure et la date courant en fournissant une valeur explicite. Cela s'applique aussi á la premiére colonne de type TIMESTAMP. Par exemple, si vous voulez affecter la date de crèation d'une ligne á une colonne de type TIMESTAMP, mais ne plus y toucher ultèrieurement :

- Laissez MySQL donner la valeur de la colonne lors de la crèation de la ligne. Cela va initialiser la colonne á la date et heure courante.
- Lorsque vous faites des modifications ultèrieures, affectez explicitement á la colonne TIMESTAMP sa propre valeur.

D'un autre cotè, vous pouvez aussi facilement initialiser la colonne TIMESTAMP avec NOW() lors de sa crèation, puis ne plus la modifier ultèrieurement.

L'intervalle de validitè des valeurs TIMESTAMP va du dèbut de l'annèe 1970 jusque quelque part durant l'annèe 2037, avec une prècision d'une seconde. Les valeurs sont affichès comme des nombres entiers.

Le format d'affichage des valeurs TIMESTAMP dèpend de la taille d'affichage, comme illustrè ci-dessous. Le format total TIMESTAMP a 14 chiffres, mais les colonnes TIMESTAMP peuvent être crèèes avec des formats plus courts :

Type de colonne	**Format d'affichage**
TIMESTAMP(14)	YYYYMMDDHHMMSS
TIMESTAMP(12)	YYMMDDHHMMSS
TIMESTAMP(10)	YYMMDDHHMM
TIMESTAMP(8)	YYYYMMDD
TIMESTAMP(6)	YYMMDD
TIMESTAMP(4)	YYMM
TIMESTAMP(2)	YY

Toutes les colonnes de type TIMESTAMP ont la même taille de stockage, indèpendamment de la taille d'affichage. Les formats les plus courants sont 6, 8, 12, et 14. Vous pouvez spècifier une taille arbitraire lors de la crèation de la table, mais 0 et les valeurs supèrieures á 14 sont ramenèes á 14. Les valeurs impaires sont aussi ramenèes au nombre pair supèrieur.

Vous pouvez spècifier les valeurs des colonnes DATETIME, DATE et TIMESTAMP, avec les formats communs suivants :

- Une chaîne au format `'AAAA-MM-JJ HH:MM:SS'` ou `'AA-MM-JJ HH:MM:SS'`. Une syntaxe "relaxèe" est premise : tout caractére de ponctuation peut être utilisè comme dèlimiteur entre les parties de temps ou heure. Par exemple, `'98-12-31 11:30:45'`, `'98.12.31 11+30+45'`, `'98/12/31 11*30*45'`, et `'98@12@31 11^30^45'` sont èquivalents.
- Une chaîne au format '`AAAA-MM-JJ HH:MM:SS`' ou '`AA-MM-JJ HH:MM:SS`'. Une syntaxe plus sympas est aussi permise : vous pouvez utiliser n'importe quel caractére de ponctuation comme dèlimiteur de date et d'heure. Par exemple, '`98/12/31 11*30*45`', et '`98@12@31 11^30^45`' sont èquivalents.
- Une chaîne au format '`AAAA-MM-JJ`' ou '`AA-MM-JJ`'. Une syntaxe plus sympas est aussi acceptèe ici. Par exemple, '`98-12-31`', '`98.12.31`', '`98/12/31`', et '`98@12@31`' sont èquivalent.
- Une chaîne sans aucun dèlimiteurs sous la forme '`AAAAMMJJHHMMSS`' ou '`AAMMJJHHMMSS`', en supposant qu'une telle chaîne ait un sens en terme de date. Par exemple '`19970523091528`' et '`970523091528`' sont interprètès comme '`1997-05-23 09:15:28`', mais '`971122129015`' est invalide (les minutes ne sont pas valides) et devient alors '`0000-00-00 00:00:00`'.
- Une chaîne sans aucun dèlimiteurs sous la forme '`AAAAMMJJ`' ou '`AAMMJJ`', en supposant qu'une telle chaîne ait un sens en terme de date. Par exemple, '`19970523`' et '`970523`' sont interprètès comme '`1997-05-23`', mais '`971332`' est invalide (les mois ne sont pas valides) et devient alors '`0000-00-00`'.
- Un nombre au format `AAAAMMJJHHMMSS` ou `AAMMJJHHMMSS`, en supposant qu'un tel nombre ait un sens en terme de date. Par exemple, `19830905132800` et `830905132800` sont interprètès comme '`1983-09-05 13:28:00`'.
- Un nombre au format `AAAAMMJJ` ou `AAMMJJ` en supposant qu'un tel nombre ait un sens en terme de date. Par exemple, `19830905` et `830905` sont interprètès comme '`1983-09-05`'.
- Un rèsultat de fonction qui retourne une valeur acceptable dans une colonne de type `DATETIME`, `DATE`, ou `TIMESTAMP`, tels que `NOW()` ou `CURRENT_DATE`.

Les valeurs invalides `DATETIME`, `DATE`, ou `TIMESTAMP` sont remplacèes par la date "zèro" du type appropriè (respectivement '`0000-00-00 00:00:00`', '`0000-00-00`', ou `00000000000000`).

Pour la valeurs spècifièes sous forme de chaînes avec des dèlimiteurs de date, il n'est pas nècessaire de spècifier les deux chiffres pour les mois ou les dates qui sont infèrieurs á `10`. Par exemple, '`1979-6-9`' est valide et est èquivalent á '`1979-06-09`'. Similairement, pour les valeurs spècifièes sous forme de chaîne avec des dèlimiteurs d'heure, il n'est pas obligatoire de spècifier les deux chiffres des heures, minutes et secondes qui sont infèrieures á `10`. '`1979-10-30 1:2:3`' est valide et est èquivalent á '`1979-10-30 01:02:03`'.

Les valeurs spècifièes sous forme de nombres doivent avoir 6, 8, 12, ou 14 chiffres de long. Si le nombre a 8 ou 14 chiffres, MySQL suppose que le format est `AAAAMMJJ` ou `AAAAMMJJHHMMSS` (respectivement) et que l'annèe est reprèsentèes par les 4 premiers chiffres. Si le nombre a 6 ou 12 chiffres, MySQL suppose que le format est `AAMMJJ` ou `AAMMJJHHMMSS` (respectivement) et format et que l'annèe est reprèsentèes par les 2 premiers chiffres. Les nombres qui ne sont pas d'une taille valide, sont complètès avec des 0 jusqu'á la taille lisible la plus proche. Les valeurs spècifièes sous forme de chaînes sans dèlimiteurs sont interprètès en fonction de

leur taille. Si la chaîne á 8 ou 14 caractéres de long, l'annèe est supposèe avoir 4 chiffres. Sinon, l'annèe est supposèe avoir 2 chiffres. La chaîne est interprètèe de gauche á droite, en lisant successivement l'annèe, le mois, la date, l'heure, les minutes et les secondes, tant qu'il y a des valeurs dans la chaîne. Cela signifie que vous ne devez pas utiliser de chaînes qui ont moins de 6 caractéres. Par exemple, si vous spècifiez '9903', en pensant qu'il reprèsente Mars 1999, vous vous apercevrez que MySQL insére á la place la date "zèro" dans votre table. Cela est dû au fait que si l'annèe et le mois sont 99 et 03, la date est 0, ce qui en fait une date invalide, qui est rejetèe par MySQL.

Les colonnes TIMESTAMP stocke des valeurs lègales en utilisant la prècision maximum fournie, indèpendamment de la taille d'affichage. Cela a plusieurs implications :

- Spècifiez toujours l'annèe, le mois et le jour, même si le type de colonne est TIMESTAMP(4) ou TIMESTAMP(2). Sinon, la valeur ne sera pas lègale et 0 sera stockèe.
- Si vous utilisez la commande ALTER TABLE pour rèduire la largeur d'une colonne TIMESTAMP, les informations qui ètaient affichèes sont dèsormais "cachèes", mais pas dètruites.
- Similairement, rèduire une colonne de type TIMESTAMP ne cause aucune perte d'information, en dehors du fait que ces informations ne sont plus affichèes.
- Bien que les valeurs TIMESTAMP soient stockèes avec une prècision d'une seconde, la seule fonction qui travaille directement avec ces valeurs est la fonction UNIX_TIMESTAMP(). Les autres fonctions opérent sur des valeurs lues et formatèes. Cela signifie que vous ne pouvez pas utiliser de fonctions telles que HOUR() ou SECOND() a moins que le format d'affichage de la valeur TIMESTAMP ne prèsente cette valeur. Par exemple, les heures ne sont jamais affichèes dans une colonne de type TIMESTAMP á moins que la taille d'affichage de la colonne ne soit d'au moins 10. L'utilisation de la fonction HOUR() sur une valeur ayant un format d'affichage plus court que 10 retournera un rèsultat inutilisable.

Dans une certaines mesure, vous pouvez assigner des valeurs d'une colonne á une autre colonne d'un autre type. Cependant, vous devez vous attendre á quelques altèration ou pertes de valeurs durant la conversion :

- Si vous assignez une valeur DATE á une colonne de type DATETIME ou TIMESTAMP, la partie reprèsentant les heures vaudra '00:00:00', car les colonnes de type DATE ne contiennent pas d'information d'heure.
- Si vous assignez une valeur DATETIME ou TIMESTAMP á une colonne de type DATE, la composante heure sera perdue, car les colonnes de type DATE ne contiennent pas d'information d'heure.
- N'oubliez pas que même si les valeurs DATETIME, DATE, et TIMESTAMP peuvent être spècifiée avec diffèrents formats, ces types n'ont pas les mêmes intervalle de validitè. Par exemple, les valeurs de type TIMESTAMP ne peuvent pas prendre de valeur antèrieure á 1970 ou postèrieure á 2037. Cela signifie qu'une date telle que '1968-01-01', est lègale dans les colonnes de type DATETIME, mais n'est pas valide pour les TIMESTAMP, et sera convertie en date zèro (0) si elle est assignèe á une telle colonne.

Attention á certains piéges concernant les spècifications de dates :

- La syntaxe á dèlimiteur libre peut être une source de probléme. Par exemple, une valeur telle que '10:11:12' ressemble á une heure, á cause du dèlimiteur ":", mais

avec une colonne de date, elle sera interprètèe comme la date '2010-11-12'. La valeur '10:45:15' sera convertie en '0000-00-00' car '45' n'est pas un mois valide.

- Le serveur MySQL effectue seulement la vèrification de base la validitè d'une date : jours 00-31, mois 00-12, annèes 1000-9999. N'importe quelle date qui n'est pas dans cette marge retournera 0000-00-00. Veuillez noter que ceci vous permet toujours de stocker les dates inadmissibles telles que 2002-04-31. Il permet á des applications web de stocker des donnèes d'une forme sans vèrifier plus loin. Pour s'assurer qu'une date est valide, vous devrez effectuer un test dans votre application.
- Les annèes spècifièe avec deux chiffres seulement sont ambigues, car il manque le siécle. MySQL interpréte les annèes á deux chiffres suivant ces régles :
  - Les annèes de l'intervalle 00-69 sont converties en 2000-2069.
  - Les annèes de l'intervalle 70-99 sont converties en 1970-1999.

6.2.2.3 Le type TIME

MySQL lit et affiche les colonnes de type TIME au format 'HH:MM:SS' (ou 'HHH:MM:SS' pour les grandes quantitès d'heures). Les valeurs de TIME vont de '-838:59:59' á '838:59:59'. La raison de cet intervalle de validitè si large est que les colonnes de type TIME peuvent être utilisès pour reprèsenter non seulement des heures du jour, mais aussi des durèes entre deux èvènements (ce qui peut dèpasser largement les 24 heures, ou même, être nègatif).

Vous pouvez spècifier une valeur de type TIME avec diffèrents formats :

- Une chaîne au format 'D HH:MM:SS.fraction'. (Notez que MySQL ne stockera pas la fraction d'une valeur TIME.)

  Vous pouvez aussi utiliser l'une des syntaxes alternatives suivantes : HH:MM:SS.fraction, HH:MM:SS, HH:MM, D HH:MM:SS, D HH:MM, D HH ou SS. Ici, D peut prendre des valeurs entre 0 et 33.
- Une chaîne sans dèlimiteur au format 'HHMMSS', en supposant que cela puisse avoir un sens en terme de date. Par exemple, '101112' est interprètè comme '10:11:12', mais '109712' est invalide (le nombre de minutes n'a pas de sens), et devient la date zèro : '00:00:00'.
- Un nombre au format HHMMSS, en supposant que cela puisse avoir un sens en terme de date. Par exemple, 101112 est interprètè comme '10:11:12'. Les formats alternatifs sont aussi compris : SS, MMSS, HHMMSS et HHMMSS.fraction. Notez que MySQL ne stocke pas encore les fractions de secondes.
- Le rèsultat d'une fonction qui retourne une valeur acceptable dans un contexte de valeurs TIME, comme CURRENT_TIME.

Pour les valeurs TIME spècifièes avec des dèlimiteurs, il n'est pas nècessaire de prèciser deux chiffres pour les valeurs infèrieurs á 10 pour les heures, minutes et secondes. '8:3:2' est la même chose que '08:03:02'.

Soyez soigneux lors de l'utilisation de valeurs "courtes" á une colonne de type TIME. MySQL interpréte les valeurs en supposant que les chiffres de droite reprèsentent les secondes (MySQL interpréte les valeurs TIME comme des durèes et non comme des heures d'une journèe). Par exemple, vous pouvez penser que les valeurs '11:12', '1112' et 1112 reprèsentent '11:12:00' (12 minutes aprés 11 heures), mais MySQL les interprètera comme

'00:11:12' (11 minutes, 12 secondes). Similairement, '12' et 12 reprèsentent '00:00:12'. Les valeurs de TIME dèclarèes avec des :, au contraire, sont toujours traitès comme des heures de journèe. '11:12' signifiera '11:12:00' et non pas 00:11:12

Les valeurs hors de l'intervalle de validitè de TIME mais qui sont valides sont ramenèes á la valeur maximale stockable la plus proche. Par exemple, '-850:00:00' et '850:00:00' sont respectivement converties en '-838:59:59' et '838:59:59'.

Les valeurs TIME non valides sont transformèes en date zèro '00:00:00'. Notez que comme '00:00:00' est elle-même une valeur TIME valide, vous n'aurez pas le moyen de faire la diffèrence entre une valeur '00:00:00' stockèe en connaissance de cause, et '00:00:00' stockèe á cause d'une erreur.

6.2.2.4 Le type YEAR

Le type YEAR est un type d'1 byte utilisè pour reprèsenter les annèes.

MySQL extrait et affiche la valeur de YEAR au format YYYY. L'èchelle va de 1901 á 2155.

Vous pouvez spècifier la valeur de YEAR en plusieurs formats :

- Une chaîne de quatre chiffres entre '1901' et '2155'.
- Un nombre á quatre chiffres entre 1901 et 2155.
- Une chaîne de deux chiffres entre '00' et '99'. Les valeurs entre '00' et '69' et entre '70' et '99' sont respectivement converties en valeurs YEAR comprises entre 2000 et 2069 d'une part, et 1970 et 1999 de l'autre.
- Une nombre de deux chiffres entre 1 et 99. Les valeurs entre 1 et 69 et entre 70 et 99 sont respectivement converties en valeurs YEAR comprises entre 2001 et 2069 d'une part, et 1970 et 1999 d'autre part. Notez que le rang de valeurs pour les nombres á deux chiffres est totalement diffèrent du rang pour les chaînes á deux chiffres parce que vous ne pouvez pas spècifier deux zèro directement en tant que nombre et le faire interprèter en tant que 2000. Vous **devez** le spècifier comme chaîne '0' ou '00' sinon il sera interprètè comme 0000.
- En tant que rèsultat d'une fonction retournant une valeur acceptable dans le contexte de YEAR, comme as NOW().

Les valeurs illègales pour YEAR sont converties en 0000.

6.2.3 Les types chaînes

Les types chaînes sont CHAR, VARCHAR, BLOB, TEXT, ENUM, et SET. Cette section dècrit comment ces types fonctionnent, leur besoins en espace et comment les utiliser dans vos requêtes.

Type	Taille maximale	Bytes
TINYTEXT ou TINYBLOB	2^8-1	255
TEXT ou BLOB	2^16-1 (64K-1)	65535
MEDIUMTEXT ou MEDIUMBLOB	2^24-1 (16M-1)	16777215
LONGBLOB	2^32-1 (4G-1)	4294967295

6.2.3.1 Les types CHAR et VARCHAR

Les types CHAR et VARCHAR sont similaires, mais différent dans la maniére dont ils sont stockès et rècupèrès.

La longueur d'une colonne CHAR est fixèe á la longueur que vous avez dèfini lors de la crèation de la table. La longueur peut être n'importe quelle valeur entre 1 and 255. (Dans la version 3.23 de MySQL, la longueuer est comprise entre 0 et 255.) Quand une valeur CHAR est enregistrèe, elle est complètè á droite avec des espaces jusqu'á atteindre la valeur fixèe. Quand une valeur de CHAR est lue, les espaces en trop sont retirès.

Les valeurs contenues dans les colonnes de type VARCHAR sont de tailles variables. Vous pouvez dèclarer une colonne VARCHAR pour que sa taille soit comprise entre 1 et 255, exactement comme pour les colonnes CHAR. Par contre, contrairement á CHAR, les valeurs de VARCHAR sont stockèes en utilisant autant de caractéres que nècessaire, plus un octet pour mèmoriser la longueur. Les valeurs ne sont pas complètèes. Au contraire, les espaces finaux sont supprimès avant stockage (ce qui ne fait pas partie des spècifications ANSI SQL).

Si vous assignez une chaîne de caractéres qui dèpasse la capacitè de la colonne CHAR ou VARCHAR, celle ci est tronquèe jusqu'á la taille maximale du champ.

Le tableau suivant illustre les diffèrences entre les deux types de colonnes en montrant les diffèrences entre l'enregistrement dans une colonne CHAR(4) ou VARCHAR(4) :

Valeur	CHAR(4)	Espace requis	VARCHAR(4)	Espace requis
''	' '	4 octets	''	1 octet
'ab'	'ab '	4 octets	'ab'	3 octets
'abcd'	'abcd'	4 octets	'abcd'	5 octets
'abcdefgh'	'abcd'	4 octets	'abcd'	5 octets

Les valeurs lues dans les colonnes de type CHAR(4) et VARCHAR(4) seront les mêmes dans tous les cas, car les espaces finaux sont retirès des valeurs issues de colonnes de type CHAR lors de la lecture.

Les valeurs dans les colonnes CHAR et VARCHAR sont classèes et comparèes sans tenir compte de la casse, á moins que l'attribut BINARY n'ai ètè spècifiè lors de la crèation de la table. L'attribut BINARY signifie que les valeurs sont classèes et triées en tenant compte de la casse, suivant l'ordre des caractéres ASCII de la machine ou est installè le serveur MySQL. BINARY n'affecte pas les mèthodes de lecture et de stockage des valeurs.

L'attribut BINARY se propage dans une expression : il suffit qu'une seule colonne, utilisèe dans une expression, ait l'attribut BINARY pour que toute l'expression ne tienne plus compte de la casse.

MySQL peut changer automatiquement le type d'une colonne CHAR ou VARCHAR lors de la crèation de la table. Voir Section 6.5.3.1 [Silent column changes], page 511.

6.2.3.2 Les types BLOB et TEXT

Une valeur de type BLOB est un objet binaire de grande taille, qui peut contenir une quantitè variable de donnèes. Les quatre types BLOB (TINYBLOB, BLOB, MEDIUMBLOB, et LONGBLOB) ne diffèrent que par la taille maximale de donnèes qu'ils peuvent stocker. Voir Section 6.2.6 [Storage requirements], page 435.

Les quatre types TEXT (TINYTEXT, TEXT, MEDIUMTEXT, et LONGTEXT correspondent aux types BLOB èquivalents, et ont les mêmes contraintes de stockage. Les seules diffèrences entre les colonnes de type BLOB et celles de type TEXT se situent aux niveau des tris et comparaisons : Les tris, faits sur les BLOB, contrairement á ceux faits sur les TEXT, tiennent compte de la casse. En d'autres termes, une valeur TEXT est une valeur BLOB insensible á la casse.

Si vous assignez une valeur trop grande á une colonne de type BLOB ou TEXT, la valeur sera tronquèe á la taille maximale possible.

Dans la majoritè des cas, vous pouvez considèrer une colonne de type TEXT comme une colonne de type VARCHAR, aussi grande que vous le souhaitez. De même, vous pouvez considèrer une colonne de type BLOB comme une colonne de type VARCHAR BINARY. Les seules diffèrences sont :

- Vous ne pouvez indexer les colonnes de type BLOB ou TEXT qu'á partir de la version 3.23.2 de MySQL.
- Il n'y a pas de suppression des espaces finaux lors du stockage de valeur dans des colonnes de type BLOB et TEXT, ce qui est le cas dans pour les colonnes de type VARCHAR.
- Les colonnes BLOB et TEXT ne peuvent avoir de valeur par dèfaut. (DEFAULT)

MyODBC considére les valeurs BLOB comme des LONGVARBINARY et les valeurs TEXT comme des LONGVARCHAR.

Vous pouvez rencontrer les problémes suivants, á cause de la grande taille des colonnes de type BLOB et TEXT, lors de leur utilisation :

- Si vous voulez utiliser les commandes GROUP BY ou ORDER BY sur une colonne de type BLOB ou TEXT, vous devez d'abord la convertir en un objet de taille fixe. Le meilleur moyen est d'utiliser la fonction SUBSTRING. Par exemple :

```
mysql> SELECT comment FROM nom_de_table,SUBSTRING(comment,20) AS substr
    ->                 ORDER BY substr;
```

  Si vous le ne faites pas, seuls les max_sort_length premiers octets de la colonne seront utilisès pour le tri. La valeur par dèfaut de max_sort_length est 1024. Cette valeur peut être modifiée en utilisant l'option -O au dèmarrage du serveur mysqld. Vous pouvez utiliser la commande GROUP BY sur une colonne de type BLOB ou TEXT en spècifiant la position de la colonne, ou avec un alias :

```
mysql> SELECT id,SUBSTRING(blob_col,1,100) FROM nom_de_table GROUP BY 2;
mysql> SELECT id,SUBSTRING(blob_col,1,100) AS b FROM nom_de_table GROUP BY b;
```

- La taille maximale d'un objet BLOB ou TEXT est dèterminèe par son type, mais la valeur la plus grande que vous pouvez transmettre au programme client est dèterminèe par la quantitè de mèmoire disponible sur le serveur et par les tailles des buffers de communication. Vous pouvez changer la taille des buffers de communication, mais vous devez le faire sur le serveur et le client en même temps. Voir Section 5.5.2 [Server parameters], page 390.

Notez que chaque valeur BLOB ou TEXT est reprèsentèe en interne par un objet allouè sèparèment, contrairement á tous les autres types de colonne, pour lesquels la place de stockage est allouèe une fois pour chaque colonne, lorsque la table est ouverte.

6.2.3.3 Le type ENUM

Une ènumèration ENUM est une chaîne dont la valeur est choisie parmi une liste de valeurs autorisèes lors de la crèation de la table.

Cette chaîne peut aussi être la chaîne vide ("") ou NULL dans certaines circonstances :

- Si vous insèrez une valeur illègale dans une ènumèration ENUM (c'est á dire, une chaîne qui n'est pas dans la liste de valeurs autorisèes), la chaîne vide est insèrèe pour reprèsenter une erreur. Cette chaîne peut être distinguèe d'une chaîne vide 'normale' par le fait que cette chaîne á la valeur numèrique 0. Nous reviendrons sur ce point plus tard.
- Si une colonne d'ènumèration est dèclarèe NULL, NULL devient aussi une valeur autorisèe, et la valeur par dèfaut est alors NULL. Si une colonne d'ènumèration est dèclarèe NOT NULL, la valeur par dèfaut est le premier èlèment de la liste des valeurs autorisèes.

Chaque èlèment de l'ènumèration dispose d'un index :

- Les valeurs de la liste des valeurs autorisèes sont indexèes á partir de 1.
- L'index de la chaîne vide (cas d'erreur) est 0. Cela signifie que vous pouvez utiliser la sèlection suivante pour repèrer les valeurs d'ènumèration invalides :

```
mysql> SELECT * FROM nom_de_table WHERE enum_col=0;
```

- L'index de la valeur NULL est NULL.

Par exemple, une colonne crèèe comme ENUM("un", "deux", "trois") peut prendre n'importe quelle valeur ci-dessous. L'index de chaque valeur est aussi prèsentè :

Valeur	Index
NULL	NULL
""	0
"un"	1
"deux"	2
"trois"	3

Une ènumèration peut avoir un maximum de 65535 èlèments.

A partir de la version 3.23.51, les espaces en dèbut et fin de chaîne sont automatiquement supprimès des èlèments de l'ènumèration ENUM lorsque la table est crèèe.

La casse des lettres est sans importance lors de l'assignation de valeurs dans une ènumèration. Cependant, les valeurs lues dans la base auront la même casse que celle spècifièe lors de la crèation de la table.

Si vous lisez le contenu d'une ènumèration dans un contexte numèrique, l'index de la valeur ENUM sera retournèe. Par exemple, vous pouvez lire des valeurs numèriques comme ceci :

```
mysql> SELECT enum_col+0 FROM nom_de_table;
```

Si vous stockez un nombre dans une colonne de type ENUM, le nombre sera traitè comme un index, et la valeur stockèe sera celle de l'èlèment ayant cet index (Attention, cela ne fonctionnera pas avec les commandes LOAD DATA, car cette derniére traite toutes les valeurs comme des chaînes). Il est dèconseillè de stocker des valeurs numèriques dans un ENUM car cela engendre des confusions.

Les valeurs de type ENUM sont triées en fonction de l'ordre des èlèments, fixè á la crèation de la table (en d'autres termes, les valeurs ENUM sont stockèes en fonction de leur index).

Par exemple, "a" prècéde "b" dans l'ènumèration ENUM("a", "b"), mais "b" prècéde "a" dans l'ènumèration ENUM("b", "a"). La chaîne vide prècéde toujours les chaînes non vides, et NULL prècéde toutes les valeurs.

Si vous voulez connaître toutes les valeurs possibles d'une colonne de type ENUM, pensez á utiliser cette commande : SHOW COLUMNS FROM nom_de_table LIKE enum_column_name, puis analysez la dèfinition de la colonne de type ENUM (deuxiéme colonne dans le rèsultat).

6.2.3.4 Le type SET

Un SET est une chaîne qui peut avoir zèro ou plusieurs valeurs, chacune doit être choisie dans une liste de valeurs dèfinies lors de la crèation de la table. Les valeurs des colonnes SET composèes de plusieurs membres sont dèfinies en sèparant celles-ci avec des virgules (','). Ce qui fait que la valeur d'un membre de SET ne peut contenir lui même de virgule.

Par exemple, une colonne dèfinie en tant que SET("un", "deux") NOT NULL peut avoir l'une de ces valeurs :

```
""
"un"
"deux"
"un,deux"
```

Un SET peut avoir au plus 64 membres.

A partir de la version 3.23.51, les espaces en trop sont automatiquement effacès des membres de SET lorsque la table est crèèe.

MySQL enregistre les valeurs de SET numèriquement. Le bit de poids faible de la valeur correspond alors au premier èlèment de la liste. Si vous utilisez une valeur SET dans un contexte numèrique, les bits des èlèments dans cet ensemble seront mis á un, et les autres á zèro. Par exemple, vous pouvez obtenir un entier á partir d'un ensemble comme ceci :

```
mysql> SELECT col_set+0 FROM nom_de_table;
```

Si un nombre est enregistrè dans une colonne SET, les bits un á un de ce nombre reprèsenteront les èlèments placès dans cet ensemble. Supposons qu'une colonne est spècifièe en tant que SET("a","b","c","d"), les membres ont alors les valeurs suivantes :

SET membre	Valeur dècimale	Valeur binaire
a	1	0001
b	2	0010
c	4	0100
d	8	1000

Si vous assignez 9 á cette colonne, cela donne 1001 en binaire, ce qui fait que les valeurs du premier et quatriéme membres "a" et "d" sont sèlectionnès et la valeur rèsultante est "a,d".

Pour les valeurs se composant de plus d'un membre du SET, l'ordre des membres n'a pas d'importance lors des insertions. Le nombre d'occurence d'un èlèment n'importe pas non plus. Lorsque la valeur sera lue ultèrieurement, chaque èlèment n'appara&icrirc;tra qu'une seule fois, et dans l'ordre donnè á la dèclaration de la colonne. Par exemple, si une colonne est spècifièe comme SET("a","b","c","d"), alors "a,d", "d,a", et "d,a,a,d,d" seront tous reprèsentès par "a,d".

Si vous spècifiez une valeur incorrecte dans une colonne `SET`, la valeur sera ignorèe.

Les valeurs de `SET` sont triées numèriquement. La valeur `NULL` prècéde toutes les autres.

Normalement, vous exècuterez un `SELECT` sur une colonne `SET` en utilisant l'opèrateur `LIKE` ou la fonction `FIND_IN_SET()` :

```
mysql> SELECT * FROM nom_de_table WHERE set_col LIKE '%value%';
mysql> SELECT * FROM nom_de_table WHERE FIND_IN_SET('value',set_col)>0;
```

Mais ce qui suit fonctionnera aussi :

```
mysql> SELECT * FROM nom_de_table WHERE set_col = 'val1,val2';
mysql> SELECT * FROM nom_de_table WHERE set_col & 1;
```

La premiére requête cherche les lignes qui correspondent exactement. La seconde ne cherche que les lignes contenant le premier membre du set.

Si vous voulez connaître toutes les valeurs possible d'une colonne `SET`, vous devez utiliser : `SHOW COLUMNS FROM nom_de_table LIKE nom_colonne_set` et ètudier la dèfinition du `SET` dans la seconde colonne.

6.2.4 Choisir le bon type de colonne

Pour une utilisation optimale des capacitèes de stockage, essayez d'utiliser le type le plus optimal dans chaque cas. Par exemple, si une colonnes du type entier sera utilisèe pour des valeurs entre `1` et `99999`, le type `MEDIUMINT UNSIGNED` sera le plus appropriè.

La reprèsentation des valeurs monètaires est un probléme commun. Avec MySQL, vous devrez utiliser le type `DECIMAL`. Il est sauvegardè en tant que chaîne, aucune perte de prècision ne devrait avoir lieu. Si la prècision n'est pas trés importante, vous pouvez utiliser le type `DOUBLE`.

Pour une haute prècision, vous pouvez toujours transcrire en nombre dècimaux, et les enregistrer dans des `BIGINT`. Cela vous permettra d'effectuer tout vos calculs avec des entiers et de convertir á nouveau en nombre dècimaux au besoin.

6.2.5 Utilisation des types de donnèes issues d'autres SGBDR

Pou faciliter l'importation de code SQL issu d'autres systémes de gestion de bases de donnèes, MySQL convertit les types de colonnes comme le montre le tableau suivant. Cette conversion facilite l'import de structures de tables :

Other vendor type	**MySQL type**
`BINARY(NUM)`	`CHAR(NUM) BINARY`
`CHAR VARYING(NUM)`	`VARCHAR(NUM)`
`FLOAT4`	`FLOAT`
`FLOAT8`	`DOUBLE`
`INT1`	`TINYINT`
`INT2`	`SMALLINT`
`INT3`	`MEDIUMINT`
`INT4`	`INT`
`INT8`	`BIGINT`
`LONG VARBINARY`	`MEDIUMBLOB`

LONG VARCHAR	MEDIUMTEXT
MIDDLEINT	MEDIUMINT
VARBINARY(NUM)	VARCHAR(NUM) BINARY

La conversion du type de colonnes s'effectue lors de la crèation. Si vous crèez une table avec des types issus d'un autre SGBDR puis que vous exècutez la commande `DESCRIBE nom_de_table`, MySQL fournira la structure de la table en utilisant les types èquivalents.

6.2.6 Capacitès des colonnes

Les capacitès de stockage de chaque type de colonnes de MySQL sont listès par catègories.

Capacitès de stockage des colonnes numèriques

Type de colonne	Espace requis
TINYINT	1 octet
SMALLINT	2 octets
MEDIUMINT	3 octets
INT	4 octets
INTEGER	4 octets
BIGINT	8 octets
FLOAT(X)	4 if X <= 24 or 8 if 25 <= X <= 53
FLOAT	4 octets
DOUBLE	8 octets
DOUBLE PRECISION	8 octets
REAL	8 octets
DECIMAL(M,D)	M+2 octets si D > 0, M+1 octets si D = 0 (D+2, si M < D)
NUMERIC(M,D)	M+2 octets si D > 0, M+1 octets si D = 0 (D+2, si M < D)

Capacitès de stockage des colonnes de date et heure

Type de colonne	Espace requis
DATE	3 octets
DATETIME	8 octets
TIMESTAMP	4 octets
TIME	3 octets
YEAR	1 octet

Capacitès de stockage des colonnes de texte

Type de colonne	Espace requis
CHAR(M)	M octets, 1 <= M <= 255
VARCHAR(M)	L+1 octets, avec L <= M et 1 <= M <= 255
TINYBLOB, TINYTEXT	L+1 octets, avec L < 2^8
BLOB, TEXT	L+2 octets, avec L < 2^16
MEDIUMBLOB, MEDIUMTEXT	L+3 octets, avec L < 2^24
LONGBLOB, LONGTEXT	L+4 octets, avec L < 2^32

`ENUM('valeur1','valeur2',...)` 1 ou 2 octets, suivant le nombre d'èlèments de l'ènumèration (65535 au maximum)

`SET('valeur1','valeur2',...)` 1, 2, 3, 4 ou 8 octets, suivant le nombre de membres de l'ensemble (64 au maximum)

Les types `VARCHAR`, `BLOB` et `TEXT` sont de longueur variable, et l'espace disque requis dèpend de la taille rèelle de la valeur prèsente dans la colonne, (taille reprèsentèe par L dans le tableau prècèdent) et non pas de la taille maximale de la colonne. Par exemple une colonne `VARCHAR(10)` peut contenir une chaîne de 10 caractéres. L'espace requis est dans ce cas lá la longueur de la chaîne (`L`), plus 1 octet pour enregistrer la longueur de celle ci. Pour la chaîne `'abcd'`, `L` est ègal á 4 et l'espace requis est de 5 octets.

Les types `BLOB` et `TEXT` requiérent 1, 2, 3, ou 4 octets pour mèmoriser la taille de la valeur dans la colonne, suivant la longueur maximale du type. Voir Section 6.2.3.2 [BLOB], page 430.

Si une table inclus au moins une colonne de taille variable, la ligne sera de taille variable. Notez que lorsqu'une table est crèèe, MySQL peut, dans certaines circonstances, changer automatiquement une colonne de taille variable en colonne á taille fixe (et vice-versa). Voir Section 6.5.3.1 [Silent column changes], page 511.

La taille d'un `ENUM` est dètèrminèe par le nombre d'èlèments de l'ènumèration. Un octet est nècessaire pour les ènumèrations ayant jusqu'á 255 valeurs possibles. Deux octets sont nècessaires pour les ènumerations ayant jusqu'á 65535 valeurs possibles. Voir Section 6.2.3.3 [ENUM], page 432.

La taille d'un `SET` est dètèerminè par le nombre de ses membres. Si il y'en a `N`, l'objet occupe `(N+7)/8` octets, arrondis á 1, 2, 3, 4, or 8 octets. Un `SET` peut avoir au plus 64 membres. Voir Section 6.2.3.4 [SET], page 433.

6.3 Fonctions á utiliser dans les clauses `SELECT` et `WHERE`

Une sèlection ou une clause WHERE dans une requête SQL peut se former d'expressions utilisant les fonctions dècrites ci-dessous.

Une expression contenant `NULL` produira toujours la valeur`NULL` comme rèsultat. (Sauf contre-indication dans le manuel)

Note : Il ne doit pas y avoir d'espace entre le nom d'une fonction et la parenthése ouvrante la suivant. Cela aide le parseur MySQL á distinguer les appels á ces fonction des rèfèrences aux tables ou colonnes ayant le même nom qu'une fonction. Les espaces autour des arguments sont autorisès.

Vous pouvez forcer MySQL á accepter les espaces aprés les nom de fonctions grâce á l'option `--ansi` de `mysqld`, ou en utilisant l'option `CLIENT_IGNORE_SPACE` avec `mysql_connect`. Dans ce cas, toutes les fonctions dèfinies deviendront des mots strictement rèservès. Voir Section 1.7.2 [ANSI mode], page 33.

Dans un soucis de simplicitè, les affichages des rèsultats de `mysql` sont fournis sous forme abrègèe. Par exemple :

```
mysql> SELECT MOD(29,9);
1 rows in set (0.00 sec)

+-----------+
| mod(29,9) |
+-----------+
|         2 |
+-----------+
```

est affichè comme ceci :

```
mysql> SELECT MOD(29,9);
        -> 2
```

6.3.1 Opèrateurs et fonctions tout-types

6.3.1.1 Parenthéses

```
( ... )
```

Utilisez les parenthéses pour forcer l'ordre des èvaluations dans une expression. Par exemple :

```
mysql> SELECT 1+2*3;
        -> 7
mysql> SELECT (1+2)*3;
        -> 9
```

6.3.1.2 Opèrateurs de comparaison

Les opèrations de comparaison donnent comme rèsultats `1` (TRUE), `0` (FALSE), ou `NULL`. Ces fonctions fonctionnent pour les nombres comme pour les chaînes. Les nombres sont automatiquement transformès en chaînes et les chaînes en nombres si besoin en est. (comme en Perl)

MySQL effectue les comparaisons suivant les régles suivantes :

- Si l'un ou les deux arguments sont `NULL`, le rèsultat de la comparaison est `NULL`, exception faite pour l'opèrateur `<=>`.
- Si les deux arguments de la comparaison sont des chaînes, ils seront comparès en tant que chaînes.
- Si les deux arguments sont des entiers, ils sont comparès en tant qu'entiers.
- Les valeurs hexadècimales sont traitèes en tant que chaînes binaires si elles ne sont pas comparèes á un nombre.
- Si l'un des arguments est une colonne de type `TIMESTAMP` ou `DATETIME` et que l'autre est une constante, celle ci est convertie en timestamp avant que la comparaison ne s'opére. Cela est fait pour être mieux compatible avec ODBC.

- Dans tous les autres cas, les arguments sont comparès en tant que nombres á dècimale flottante. (rèels)

Par dèfaut, la comparaison des chaînes s'effectue d'une façcon insensible á la casse en utilisant le jeu de charactéres courant (ISO-8859-1 Latin1 par dèfaut, qui fonctionne aussi trés bien pour l'anglais).

Les exemples suivants, montrent la conversion des chaînes en nombres pour les opèrations de comparaison :

```
mysql> SELECT 1 > '6x';
        -> 0
mysql> SELECT 7 > '6x';
        -> 1
mysql> SELECT 0 > 'x6';
        -> 0
mysql> SELECT 0 = 'x6';
        -> 1
```

`=` Egal :

```
mysql> SELECT 1 = 0;
        -> 0
mysql> SELECT '0' = 0;
        -> 1
mysql> SELECT '0.0' = 0;
        -> 1
mysql> SELECT '0.01' = 0;
        -> 0
mysql> SELECT '.01' = 0.01;
        -> 1
```

`<>`
`!=` Diffèrent :

```
mysql> SELECT '.01' <> '0.01';
        -> 1
mysql> SELECT .01 <> '0.01';
        -> 0
mysql> SELECT 'zapp' <> 'zappp';
        -> 1
```

`<=` Infèrieur ou ègal :

```
mysql> SELECT 0.1 <= 2;
        -> 1
```

`<` Strictement infèrieur :

```
mysql> SELECT 2 < 2;
        -> 0
```

`>=` Supèrieur ou ègal :

```
mysql> SELECT 2 >= 2;
        -> 1
```

`>` Strictement supèrieur :

```
mysql> SELECT 2 > 2;
        -> 0
```

`<=>` Egalitè sure avec NULL :

```
mysql> SELECT 1 <=> 1, NULL <=> NULL, 1 <=> NULL;
        -> 1 1 0
```

`IS NULL`
`IS NOT NULL`

Tester si une valeur est ou n'est pas `NULL`:

```
mysql> SELECT 1 IS NULL, 0 IS NULL, NULL IS NULL;
        -> 0 0 1
mysql> SELECT 1 IS NOT NULL, 0 IS NOT NULL, NULL IS NOT NULL;
        -> 1 1 0
```

Pour être compatible avec les autres programmes, MySQL gére les appels qui utilisent `IS NULL` de la façon suivante :

- Vous pouvez trouver le dernier enregistrement insèrè en utilisant :

  ```
  SELECT * FROM nom_de_table WHERE auto_col IS NULL
  ```

  Cela peut être interdit en mettant `SQL_AUTO_IS_NULL=0`. Voir Section 5.5.6 [SET OPTION], page 396.

- For `NOT NULL DATE` and `DATETIME` columns you can find the special date `0000-00-00` by using:

  ```
  SELECT * FROM nom_de_table WHERE date_column IS NULL
  ```

  This is needed to get some ODBC applications to work (as ODBC doesn't support a `0000-00-00` date)

`expression BETWEEN min AND max`

Si `expression` est supèrieure ou ègale á `min` et `expression` est infèrieure ou ègale á `max`, `BETWEEN` retourne `1`, sinon `0`. Ceci est èquivalent á l'expression `(min <= expression AND expression <= max)` si tous les arguments sont du même type. Dans tous les autres cas, la conversion de type prends place, selon les régles suivantes, mais appliquèe aux trois arguments. **Notez** que avant la 4.0.5, les arguments ètaient convertis au type de `expr`.

```
mysql> SELECT 1 BETWEEN 2 AND 3;
        -> 0
mysql> SELECT 'b' BETWEEN 'a' AND 'c';
        -> 1
mysql> SELECT 2 BETWEEN 2 AND '3';
        -> 1
mysql> SELECT 2 BETWEEN 2 AND 'x-3';
        -> 0
```

`expr NOT BETWEEN min AND max`

Même chose que `NOT (expr BETWEEN min AND max)`.

`expr IN (valeur,...)`
: Retourne `1` si `expr` est l'une des valeurs dans la liste `IN`, sinon retourne `0`. Si toutes les valeurs sont des constantes, toutes les valeurs sont èvaluèes avec le type de `expr` et triées. La recherche de l'èlèment est alors faite en utilisant la recherche binaire. Cela signifie que `IN` est trés rapide si les valeurs contenues dans la liste `IN` sont toutes des constantes. Si `expr` est une chaîne sensible á la casse, la comparaison est faite dans un contexte sensible á la casse :

```
mysql> SELECT 2 IN (0,3,5,'wefwf');
        -> 0
mysql> SELECT 'wefwf' IN (0,3,5,'wefwf');
        -> 1
```

`expr NOT IN (value,...)`
: Même chose que `NOT (expr IN (valeur,...))`.

`ISNULL(expr)`
: Si `expr` est `NULL`, `ISNULL()` retourne `1`, sinon il retourne `0`:

```
mysql> SELECT ISNULL(1+1);
        -> 0
mysql> SELECT ISNULL(1/0);
        -> 1
```

Notez que la comparaison de deux valeurs `NULL` en utilisant `=` donnera toujours false !

`COALESCE(list)`
: Retourne le premier èlèment non-`NULL` de la liste :

```
mysql> SELECT COALESCE(NULL,1);
        -> 1
mysql> SELECT COALESCE(NULL,NULL,NULL);
        -> NULL
```

`INTERVAL(N,N1,N2,N3,...)`
: Retourne `0` si `N < N1`, `1` si `N < N2` etc... Tous les arguments sont traitès en tant qu'entiers. Il est requis que `N1 < N2 < N3 < ... < Nn` pour que cette fonction fonctionne correctement. Cela est due á la recherche binaire utilisèe (trés rapide) :

```
mysql> SELECT INTERVAL(23, 1, 15, 17, 30, 44, 200);
        -> 3
mysql> SELECT INTERVAL(10, 1, 10, 100, 1000);
        -> 2
mysql> SELECT INTERVAL(22, 23, 30, 44, 200);
        -> 0
```

Si vous comparez des chaînes non sensibles á la casse avec l'un des opèrateurs standards (`=`, `<>`..., mais pas `LIKE`) les espaces en dèbut et fin de chaîne (espaces, tabulations et nouvelles lignes) seront ignorèes.

```
mysql> SELECT "a" ="A \n";
        -> 1
```

6.3.1.3 Opèrateurs logiques

En SQL, tous les opèrateurs logiques èvaluent á TRUE, FALSE ou NULL (INCONNU). En MySQL, c'est implèmentè en `1` (TRUE), `0` (FALSE), et `NULL`. La plupart de ce qui suit est est commun entre les diffèrentss bases de donnèes SQL, pourtant, certains systéme pourraient retourner une valeur non nulle pour TRUE (pas obligatoirement 1).

`NOT`
`!` NOT (NON) logique. Evalue á `1` si l'opèrande est `0`, á `0` si l'opèrande est non nulle, et `NOT NULL` retourne `NULL`.

```
mysql> SELECT NOT 10;
        -> 0
mysql> SELECT NOT 0;
        -> 1
mysql> SELECT NOT NULL;
        -> NULL
mysql> SELECT ! (1+1);
        -> 0
mysql> SELECT ! 1+1;
        -> 1
```

Le dernier exemple donne `1` car l'expression est èvaluèe commes `(!1)+1`.

`AND`
`&&` AND (ET) logique. Evalue á `1` si toutes les opèrandes sont diffèrentes de zèro et de `NULL`, á `0` si l'une des opèrandes est `0`, dans les autres cas, `NULL` est retournè.

```
mysql> SELECT 1 && 1;
        -> 1
mysql> SELECT 1 && 0;
        -> 0
mysql> SELECT 1 && NULL;
        -> NULL
mysql> SELECT 0 && NULL;
        -> 0
mysql> SELECT NULL && 0;
        -> 0
```

Notez que pour les versions antèrieures á la 4.0.5 l'èvaluation est intèrrompue lorsque `NULL` est rencontrè, au lieu de continuer á tester une èventuelle existence de `0`. Cela signifie que dans ces versions, `SELECT (NULL AND 0)` retourne `NULL` au lieu de `0`. En 4.0.5 le code a ètè revu pour que le rèsultat rèponde toujours au normes ANSI tout en optimisant le plus possible.

`OR`
`||` OR (OU inclusif) logique. Evalue á `1` si aucune opèrande n'est nulle, á `NULL` si l'une des opèrandes est `NULL`, sinon `0` est retournè.

```
mysql> SELECT 1 || 1;
        -> 1
mysql> SELECT 1 || 0;
```

```
        -> 1
mysql> SELECT 0 || 0;
        -> 0
mysql> SELECT 0 || NULL;
        -> NULL
mysql> SELECT 1 || NULL;
        -> 1
```

`XOR` XOR (OU exclusif) logique. Retourne `NULL` si l'une des opèrandes est `NULL`. Pour les opèrandes non-`NULL`, èvalue á `1` si un nombre pair d'opèrandes est non-nul, sinon `0` est retournè.

```
mysql> SELECT 1 XOR 1;
        -> 0
mysql> SELECT 1 XOR 0;
        -> 1
mysql> SELECT 1 XOR NULL;
        -> NULL
mysql> SELECT 1 XOR 1 XOR 1;
        -> 1
```

`a XOR b` est mathèmatiquement ègal á `(a AND (NOT b)) OR ((NOT a) and b)`.

6.3.1.4 Les fonctions de contrôle

`IFNULL(expr1,expr2)`

Si l'argument `expr1` n'est pas `NULL`, la fonction `IFNULL()` retournera l'argument `expr1`, sinon elle retournera l'argument `expr2`. La fonction `IFNULL()` retourne une valeur numèrique ou une chaîne de caractéres, suivant le contexte d'utilisation :

```
mysql> SELECT IFNULL(1,0);
        -> 1
mysql> SELECT IFNULL(NULL,10);
        -> 10
mysql> SELECT IFNULL(1/0,10);
        -> 10
mysql> SELECT IFNULL(1/0,'oui');
        -> 'oui'
```

`NULLIF(expr1,expr2)`

Si l'expression `expr1 = expr2` est vrai, la fonction retourne `NULL` sinon elle retourne `expr1`. Cela revient á faire `CASE WHEN x = y THEN NULL ELSE x END`:

```
mysql> SELECT NULLIF(1,1);
        -> NULL
mysql> SELECT NULLIF(1,2);
        -> 1
```

Notez que l'argument `expr1` est èvaluè deux fois dans MySQL si les arguments sont ègaux.

`IF(expr1,expr2,expr3)`

Si l'arguement `expr1` vaut TRUE (`expr1 <> 0` et `expr1 <> NULL`) alors la fonction `IF()` retourne l'argument `expr2`, sinon, elle retourne l'argument `expr3`. La fonction `IF()` retourne une valeur numèrique ou une chaîne de caractéres, suivant le contexte d'utilisation :

```
mysql> SELECT IF(1>2,2,3);
        -> 3
mysql> SELECT IF(1<2,'oui','non');
        -> 'oui'
mysql> SELECT IF(STRCMP('test','test1'),'non','oui');
        -> 'non'
```

Si l'argument `expr2` ou `expr3` est explicitement `NULL` alors the le type du rèsultat de la fonction `IF()` est le type de la colonne non `NULL`. (Ce comportement est nouveau dans MySQL 4.0.3).

L'argument `expr1` est èvaluè comme un entier, cela signifie que si vous testez un nombre á virgule flottante ou une chaîne de caractéres, vous devez utiliser une opèration de comparaison :

```
mysql> SELECT IF(0.1,1,0);
        -> 0
mysql> SELECT IF(0.1<>0,1,0);
        -> 1
```

Dans le premier exemple ci-dessus, `IF(0.1)` retourne `0` parce que `0.1` est converti en une chaîne de caractéres, ce qui revient á tester `IF(0)`. Ce n'est certainement pas ce que vous dèsireriez. Dans le second exemple, la comparaison teste si le nombre á virgule flottante est diffèrent de zèro. Le rèsultat de cette comparaison sera un entier.

Le type de la fonction `IF()` (ce qui peut être important s'il est stockè dans une table temporaire) est calculè, dans la Version 3.23 de MySQL, comme suit :

Expression	**Valeur retournèe**
expr2 ou expr3 retourne une chaîne	chaîne
expr2 ou expr3 retourne un nombre á virgule	nombre á virgule
expr2 ou expr3 retourne un entier	entier

Si expr2 et expr3 sont des chaînes de caractéres, alors le rèsultat est insensible á la casse si les deux chaînes de caractéres sont insensibles á la casse. (A partir de la version 3.23.51 de MySQL)

`CASE valeur WHEN [compare-value] THEN rèsultat [WHEN [compare-value] THEN rèsultat ...] [ELSE rèsultat] END`
`CASE WHEN [condition] THEN rèsultat [WHEN [condition] THEN rèsultat ...] [ELSE rèsultat] END`

La premiére version retourne `rèsultat` si `valeur=compare-value`. La seconde version retourne le rèsultat de la premiére condition qui se rèalise. Si aucune des conditions n'est rèalisè, alors le rèsultat de la clasue `ELSE` est retournè. Si il n'y a pas de clause `ELSE` alors `NULL` est retournè :

```
mysql> SELECT CASE 1 WHEN 1 THEN "un"
           WHEN 2 THEN "deux" ELSE "plus" END;
       -> "un"
mysql> SELECT CASE WHEN 1>0 THEN "vrai" ELSE "faux" END;
       -> "vrai"
mysql> SELECT CASE BINARY "B" WHEN "a" THEN 1 WHEN "b" THEN 2 END;
       -> NULL
```

Le type de la valeur retournèe (`INTEGER`, `DOUBLE` ou `STRING`) est de même type que la premiére valeur retournèe (l'expression aprés le premier `THEN`).

6.3.2 Fonctions de chaînes de caractéres

Les fonctions qui traitent les chaînes de caractéres retournent `NULL` si la longueur du rèsultat finit par dèpasser la taille maximale du paramétre `max_allowed_packet`, dèfini dans la configuration du serveur. Voir Section 5.5.2 [Server parameters], page 390.

Pour les fonctions qui opérent sur des positions á l'intèrieur d'une chaîne, la position initiale est 0.

`ASCII(str)`

Retourne le code ASCII du premier caractére de la chaîne de caractéres `str`. Retourne `0` si la chaîne de caractére `str` est vide. Retourne `NULL` si la chaîne de caractéres `str` est `NULL` :

```
mysql> SELECT ASCII('2');
        -> 50
mysql> SELECT ASCII(2);
        -> 50
mysql> SELECT ASCII('dx');
        -> 100
```

Voir aussi la fonction `ORD()`.

`ORD(str)` Si le premier caractére de la chaîne `str` est un caractére multi-octets, la fonction retourne le code de ce caractére, calculè á partir du code ASCII retournè par cette formule : `((first byte ASCII code)*256+(second byte ASCII code))[*256+third byte ASCII code...]`. Si le premier caractére n'est pas un caractére multi-octet, la fonction retournera la même valeur que la fonction `ASCII()` :

```
mysql> SELECT ORD('2');
        -> 50
```

`CONV(N,from_base,to_base)`

Convertit des nombres entre diffèrentes bases. Retourne une chaîne de caractéres reprèsentant le nombre `N`, convertit de la base `from_base` vers la base `to_base`. La fonction retourne `NULL` si un des arguments est `NULL`. L'argument `N` est interprètè comme un entier, mais peut être spècifiè comme un entier ou une chaîne de caractéres. Le minimum pour la base est `2` et son maximum est `36`. Si `to_base` est un nombre nègatif, `N` sera considèrè comme un nombre signè.

Dans le cas contraire, N sera traitè comme un nombre non-signè. La fonction CONV travaille avec une prècision de 64 bits :

```
mysql> SELECT CONV("a",16,2);
        -> '1010'
mysql> SELECT CONV("6E",18,8);
        -> '172'
mysql> SELECT CONV(-17,10,-18);
        -> '-H'
mysql> SELECT CONV(10+"10"+'10'+0xa,10,10);
        -> '40'
```

BIN(N) Retourne une chaîne de caractéres reprèsentant la valeur binaire de l'argument N, oú l'argument N est un nombre de type BIGINT. Cette fonction est un èquivalent de CONV(N,10,2). Retourne NULL si l'argument N est NULL :

```
mysql> SELECT BIN(12);
        -> '1100'
```

OCT(N) Retourne une chaîne de caractéres reprèsentant la valeur octal de l'argument N, oú l'argument N est un nombre de type BIGINT. Cette fonction est un èquivalent de CONV(N,10,8). Retourne NULL si l'argument N est NULL:

```
mysql> SELECT OCT(12);
        -> '14'
```

HEX(N_or_S)

Si l'argument N_OR_S est un nombre, cette fonction retournera une chaîne de caractére reprèsentant la valeur hexadècimale de l'argument N, oú l'argument N est de type BIGINT. Cette fonction est un èquivalent de CONV(N,10,16).

Si N_OR_S est une chaîne de caractéres, cette fonction retournera une chaîne de caractéres hexadècimale de N_OR_S oú chaque caractére de N_OR_S est converti en 2 chiffres hexadècimaux. C'est l'inverse de la chaîne 0xff.

```
mysql> SELECT HEX(255);
        -> 'FF'
mysql> SELECT HEX("abc");
        -> 616263
mysql> SELECT 0x616263;
        -> "abc"
```

CHAR(N,...)

La fonction CHAR() interpréte les arguments comme des entiers et retourne une chaîne de caractéres, constituèe des caractéres, identifiès par leur code ASCII. Les valeurs NULL sont ignorèes :

```
mysql> SELECT CHAR(77,121,83,81,'76');
        -> 'MySQL'
mysql> SELECT CHAR(77,77.3,'77.3');
        -> 'MMM'
```

CONCAT(str1,str2,...)

Retourne une chaîne reprèsentant la concatènation des arguments. Retourne NULL si un des arguments est NULL. Cette fonction peut prendre plus de 2

arguments. Si un argument est un nombre, il sera converti en son èquivalent sous forme de chaîne de caractéres :

```
mysql> SELECT CONCAT('My', 'S', 'QL');
        -> 'MySQL'
mysql> SELECT CONCAT('My', NULL, 'QL');
        -> NULL
mysql> SELECT CONCAT(14.3);
        -> '14.3'
```

`CONCAT_WS(separator, str1, str2,...)`

La fonction `CONCAT_WS()` signifie `CONCAT With Separator`, c'est-á-dire "concatènation avec sèparateur. Le premier argument est le sèparateur utilisè pour le reste des arguments. Le sèparateur peut être une chaîne de caractéres, tout comme le reste des arguments. Si le sèparateur est `NULL`, le rèsultat sera `NULL`. Cette fonction ignorera tous les arguments de valeur `NULL` et vides, hormis le sèparateur. Le sèparateur sera ajoutè entre tous les arguments á concatèner :

```
mysql> SELECT CONCAT_WS(",","Premier nom","Deuxiéme nom","Dernier nom");
       -> 'Premier nom,Deuxiéme nom,Dernier nom'
mysql> SELECT CONCAT_WS(",","Premier nom",NULL,"Dernier nom");
       -> 'Premier nom,Dernier nom'
```

`LENGTH(str)`
`OCTET_LENGTH(str)`
`CHAR_LENGTH(str)`
`CHARACTER_LENGTH(str)`

Retourne le nombre de caractéres de la chaîne `str`:

```
mysql> SELECT LENGTH('text');
        -> 4
mysql> SELECT OCTET_LENGTH('text');
        -> 4
```

Notez que pour la fonction `CHAR_LENGTH()` et la fonction `CHARACTER_LENGTH()`, les caractéres multi-octets sont comptès une fois seulement.

`BIT_LENGTH(str)`

Retourne le nombre de bits de la chaîne de caractéres `str` :

```
mysql> SELECT BIT_LENGTH('text');
        -> 32
```

`LOCATE(substr,str)`
`POSITION(substr IN str)`

Retourne la position de la premiére occurrence de la chaîne `substr` dans la chaîne de caractéres `str`. Retourne `0` si `substr` ne se trouve pas dans la chaîne de caractéres `str`:

```
mysql> SELECT LOCATE('bar', 'foobarbar');
        -> 4
mysql> SELECT LOCATE('xbar', 'foobar');
        -> 0
```

Cette fonction gére les caractéres multi-octets. Dans la version 3.23 de MySQL, cette fonction est sensible á la casse, alors que dans la version 4.0 de MySQL, cette fonction sera sensible á la casse si l'argument est une chaîne de caractéres binaire.

`LOCATE(substr,str,pos)`

Retourne la position de la premiére occurrence de la chaîne `substr` dans la chaîne de caractéres `str`, á partir de la position `pos`. Retourne `0` si `substr` ne se trouve pas dans la chaîne de caractéres `str`:

```
mysql> SELECT LOCATE('bar', 'foobarbar',5);
        -> 7
```

Cette fonction gére les caractéres multi-octets. Dans la version 3.23 de MySQL, cette fonction est sensible á la casse, alors que dans la version 4.0 de MySQL, cette fonction sera sensible á la casse si l'argument est une chaîne de caractéres binaire.

`INSTR(str,substr)`

Retourne la position de la premiére occurence de la chaîne `substr` dans la chaîne de caractéres `str`. Cette fonction est exactement la même que la fonction `LOCATE()`, á la diffèrence que ces arguments sont inversès :

```
mysql> SELECT INSTR('foobarbar', 'bar');
        -> 4
mysql> SELECT INSTR('xbar', 'foobar');
        -> 0
```

Cette fonction gére les caractéres multi-octets. Dans la version 3.23 de MySQL, cette fonction est sensible á la casse, alors que dans la version 4.0 de MySQL, cette fonction sera sensible á la casse si l'argument est une chaîne de caractéres binaire.

`LPAD(str,len,padstr)`

Retourne la chaîne de caractéres `str`, complètèe á gauche par la chaîne de caractéres `padstr` jusqu'á ce que la chaîne de caractéres `str` atteigne `len` caractéres de long. Si la chaîne de caractéres `str` est plus longue que `len'` caractéres, elle sera raccourcie de `len` caractéres.

```
mysql> SELECT LPAD('hi',4,'??');
        -> '??hi'
```

`RPAD(str,len,padstr)`

Retourne la chaîne de caractéres `str`, complètèe á droite par la chaîne de caractéres `padstr` jusqu'á ce que la chaîne de caractéres `str` atteigne `len` caractéres de long. Si la chaîne de caractéres `str` est plus longue que `len'` caractéres, elle sera raccourcie de `len` caractéres.

```
mysql> SELECT RPAD('hi',5,'?');
        -> 'hi???'
```

`LEFT(str,len)`

Retourne les `len` caractéres les plus á gauche de la chaîne de caractéres `str` :

```
mysql> SELECT LEFT('foobarbar', 5);
```

```
        -> 'fooba'
```

Cette fonction gére les caractéres multi-octets.

`RIGHT(str,len)`

Retourne les `len` caractéres les plus á droite de la chaîne de caractéres `str` :

```
mysql> SELECT RIGHT('foobarbar', 4);
        -> 'rbar'
```

Cette fonction gére les caractéres multi-octets.

`SUBSTRING(str,pos,len)`
`SUBSTRING(str FROM pos FOR len)`
`MID(str,pos,len)`

Retourne une chaîne de `len` caractéres de long dela chaîne `str`, á partir de la position `pos`. La syntaxe ANSI SQL92 utilise une variante de la fonction `FROM` :

```
mysql> SELECT SUBSTRING('Quadratically',5,6);
        -> 'ratica'
```

Cette fonction gére les caractéres multi-octets.

`SUBSTRING(str,pos)`
`SUBSTRING(str FROM pos)`

Retourne une portion de la chaîne de caractéres `str` á partir de la position `pos` :

```
mysql> SELECT SUBSTRING('Quadratically',5);
        -> 'ratically'
mysql> SELECT SUBSTRING('foobarbar' FROM 4);
        -> 'barbar'
```

Cette fonction gére les caractéres multi-octets.

`SUBSTRING_INDEX(str,delim,count)`

Retourne une portion de la chaîne de caractéres `str`, située avant `count` occurences du dèlimiteur `delim`. Si l'argument `count` est positif, tout ce qui prècéde le dèlimiteur final sera retournè. Si l'argument `count` est nègatif, tout ce qui prècéde le dèlimiteur final sera retournè :

```
mysql> SELECT SUBSTRING_INDEX('www.mysql.com', '.', 2);
        -> 'www.mysql'
mysql> SELECT SUBSTRING_INDEX('www.mysql.com', '.', -2);
        -> 'mysql.com'
```

Cette fonction gére les caractéres multi-octets.

`LTRIM(str)`

Retourne la chaîne de caractéres `str` sans les espaces initiaux :

```
mysql> SELECT LTRIM('  barbar');
        -> 'barbar'
```

`RTRIM(str)`

Retourne la chaîne de caractéres `str` sans les espaces finaux :

```
mysql> SELECT RTRIM('barbar   ');
        -> 'barbar'
```

Cette fonction gére les caractéres multi-octets.

`TRIM([[BOTH | LEADING | TRAILING] [remstr] FROM] str)`
: Retourne la chaîne de caractéres `str` dont tous les prèfixes et/ou suffixes `remstr` ont ètè supprimès. Si aucun des spècificateurs `BOTH`, `LEADING` ou `TRAILING` sont fournis, `BOTH` est utilisè comme valeur par dèfaut. Si `remstr` n'est pas spècifiè, les espaces sont supprimès :

```
mysql> SELECT TRIM('  bar   ');
        -> 'bar'
mysql> SELECT TRIM(LEADING 'x' FROM 'xxxbarxxx');
        -> 'barxxx'
mysql> SELECT TRIM(BOTH 'x' FROM 'xxxbarxxx');
        -> 'bar'
mysql> SELECT TRIM(TRAILING 'xyz' FROM 'barxxyz');
        -> 'barx'
```

Cette fonction gére les caractéres multi-octets.

`SOUNDEX(str)`
: Retourne la valeur Soundex de la chaîne de caractéres `str`. Deux chaînes qui ont des sonoritès proches auront des valeurs soundex proches. Une chaîne Soundex standart posséde 4 caractéres, mais la fonction `SOUNDEX()` retourne une chaîne de longueur arbitraire. Vous pouvez utiliser la fonction `SUBSTRING()` sur ce rèsultat pour obtenir une chaine Soundex standart. Tout caractére non alpha-numèrique sera ignorè. Tous les caractéres internationaux qui ne font pas partie de l'alphabet de base (A-Z) seront considèrès comme des voyelles :

```
mysql> SELECT SOUNDEX('Hello');
        -> 'H400'
mysql> SELECT SOUNDEX('Quadratically');
        -> 'Q36324'
```

`SPACE(N)` Retourne une chaîne constituèe de `N` espaces :

```
mysql> SELECT SPACE(6);
        -> '      '
```

`REPLACE(str,from_str,to_str)`
: Retourne une chaîne de caractéres `str` dont toutes les occurences de la chaîne `from_str` sont remplacèes par la chaîne `to_str` :

```
mysql> SELECT REPLACE('www.mysql.com', 'w', 'Ww');
        -> 'WwWwWw.mysql.com'
```

Cette fonction gére les caractéres multi-octets.

`REPEAT(str,count)`
: Retourne une chaîne de caractéres constituèe de la rèpètition de `count` fois la chaîne `str`. Si `count <= 0`, retourne une chaîne vide. Retourne `NULL` si `str` ou `count` sont `NULL` :

```
mysql> SELECT REPEAT('MySQL', 3);
```

```
        -> 'MySQLMySQLMySQL'
```

`REVERSE(str)`

Retourne une chaîne dont l'ordre des caractéres est l'inverse de la chaîne `str` :

```
mysql> SELECT REVERSE('abc');
        -> 'cba'
```

Cette fonction gére les caractéres multi-octets.

`INSERT(str,pos,len,newstr)`

Retourne une chaîne de caractéres `str`, aprés avoir remplacè la portion de chaîne commençant á la position `pos` et de longueur `len` caractéres, par la chaîne `newstr` :

```
mysql> SELECT INSERT('Quadratic', 3, 4, 'What');
        -> 'QuWhattic'
```

Cette fonction gére les caractéres multi-octets.

`ELT(N,str1,str2,str3,...)`

Retourne `str1` si `N = 1`, `str2` si `N = 2`, et ainsi de suite. Retourne `NULL` si `N` est plus petit que `1` ou plus grand que le nombre d'arguments. La fonction `ELT()` est un complèment de la fonction `FIELD()` :

```
mysql> SELECT ELT(1, 'ej', 'Heja', 'hej', 'foo');
        -> 'ej'
mysql> SELECT ELT(4, 'ej', 'Heja', 'hej', 'foo');
        -> 'foo'
```

`FIELD(str,str1,str2,str3,...)`

Retourne l'index de la chaîne `str` dans la liste `str1`, `str2`, `str3`, ... Retourne `0` si `str` n'est pas trouvè. La fonction `FIELD()` est un complèment de la fonction `ELT()`:

```
mysql> SELECT FIELD('ej', 'Hej', 'ej', 'Heja', 'hej', 'foo');
        -> 2
mysql> SELECT FIELD('fo', 'Hej', 'ej', 'Heja', 'hej', 'foo');
        -> 0
```

`FIND_IN_SET(str,strlist)`

Retourne une valeur de `1` á `N` si la chaîne `str` se trouve dans la liste `strlist` constituèe de `N` chaînes. Une liste de chaîne est une chaîne composèe de sous-chaînes sèparèes par une virgule ‘`,`’. Si le premier argument est une chaîne constante et le second, une colonne de type `SET`, la fonction `FIND_IN_SET()` est optimisèe pour utiliser une recherche binaire trés rapide. Retourne `0` si `str` n'est pas trouvè dans la liste `strlist` ou si la liste `strlist` est une chaîne vide. Retourne `NULL` si l'un des arguments est `NULL`. Cette fonction ne fonctionne pas correctement si le premier argument contient une virgule ‘`,`’ :

```
mysql> SELECT FIND_IN_SET('b','a,b,c,d');
        -> 2
```

`MAKE_SET(bits,str1,str2,...)`

Retourne une liste (une chaîne contenant des sous-chaînes sèparèes par une virgule ‘`,`’) constituèe de chaînes qui ont le bit correspondant dans la liste

bits. str1 correspond au bit 0, str2 au bit 1, etc... Les chaines NULL dans les listes str1, str2, ... sont ignorèes :

```
mysql> SELECT MAKE_SET(1,'a','b','c');
        -> 'a'
mysql> SELECT MAKE_SET(1 | 4,'hello','nice','world');
        -> 'hello,world'
mysql> SELECT MAKE_SET(0,'a','b','c');
        -> ''
```

EXPORT_SET(bits,on,off,[sèparateur,[nombre_de_bits]])

Retourne une chaîne dont tous les bits á 1 dans 'bit' sont reprèsentès par la chaîne 'on', et dont tous les bits á 0 sont reprèsentès par la chaîne 'off'. Chaque chaîne est sèparèe par 'sèparateur' (par dèfaut, une virgule ',') et seul 'nombre_de_bits' (par dèfaut, 64) 'bits' est utilisè :

```
mysql> SELECT EXPORT_SET(5,'Y','N',',',4)
        -> Y,N,Y,N
```

LCASE(str)
LOWER(str)

Retourne une chaîne str dont tous les caractéres ont ètè mis en minuscule, en accord avec le charset courant (le charset par dèfaut est ISO-8859-1 Latin1) :

```
mysql> SELECT LCASE('QUADRATICALLY');
        -> 'quadratically'
```

Cette fonction gére les caractéres multi-octets.

UCASE(str)
UPPER(str)

Retourne une chaîne str dont tous les caractéres ont ètè mis en majuscule, en accord avec le charset courant (le charset par dèfaut est ISO-8859-1 Latin1) :

```
mysql> SELECT UCASE('Hej');
        -> 'HEJ'
```

Cette fonction gére les caractéres multi-octets.

LOAD_FILE(file_name)

Lit le fichier file_name et retourne son contenu sous la forme d'une chaîne de caractéres. Le fichier doit se trouver sur le serveur qui exècute MySQL, vous devez spècifier le chemin absolu du fichier et vous devezavoir les droits en lecture sur celui-ci. Le fichier doit pouvoir être lisible par tous et doit être plus petit que max_allowed_packet.

Si ce fichier n'existe pas ou ne peut pas être lu pour diffèrentes raisons, la fonction retourne NULL :

```
mysql> UPDATE tbl_name
           SET blob_column=LOAD_FILE("/tmp/picture")
           WHERE id=1;
```

Si vous n'utilisez pas la version 3.23 de MySQL, vous devez lire le fichier depuis votre application et crèer ainsi votre requête INSERT vous-même, pour mettre á jour la base de donnèes avec le contenu de ce fichier. Une des possibilitès

pour rèaliser ceci, si vous utilisez la librairie MySQL++, peut être trouvèe á `http://www.mysql.com/documentation/mysql++/mysql++-examples.html`.

`QUOTE(str)`

Echappe les caractéres d'une chaîne pour produire un rèsultat qui sera exploitable dans une requête SQL. Les caractéres suivants seront prècèdès d'un anti-slash dans la chaîne retournèe : le guillement simple (‘’’), l'anti-slash (‘\’), ASCII NUL, et le Control-Z. Si l'argument vaut `NULL`, la valeur retournèe sera le mot “NULL” sans les guillements simples.

```
mysql> SELECT QUOTE("Don't");
        -> 'Don\'t!'
mysql> SELECT QUOTE(NULL);
        -> NULL
```

MySQL convertit automatiquement les nombres en une chaînes de caractéres, si nècessaire et vice-versa :

```
mysql> SELECT 1+"1";
        -> 2
mysql> SELECT CONCAT(2,' test');
        -> '2 test'
```

Si vous voulez convertir explicitement un nombre en une chaîne de caractéres, passez-le en argument á la fonction `CONCAT()`.

Si une fonction de chaînes de caractéres prend en argument une chaîne binaire, le rèsultat sera ègalement une chaîne binaire. Un nombre converti en une chaîne sera traitè comme ètant une chaîne binaire. Celá affecte uniquement les comparaisons.

6.3.2.1 Opèrateurs de comparaison pour les chaînes de caractéres

Normalement, si l'une des expressions dans une comparaison de chaîne est sensible á la casse, la comparaison est exècutèe en tenant compte de la casse.

`expr LIKE pat [ESCAPE 'escape-char']`

La rèalisation d'expression utilisant les expressions règuliéres simples de comparaison de SQL. Retourne `1` (TRUE) ou `0` (FALSE). Avec `LIKE`, vous pouvez utiliser les deux jokers suivants :

Char	Description
%	Remplace n'importe quel nombre de caractéres, y compris aucun
_	Remplace exactement un caractére

```
mysql> SELECT 'David!' LIKE 'David_';
        -> 1
mysql> SELECT 'David!' LIKE '%D%v%';
        -> 1
```

Pour tester la prèsence littèrale d'un joker, prècèdez-le d'un caractére d'èchappement. Si vous ne spècifiez pas le caractére d'èchappement `ESCAPE`, le caractére ‘\’ sera utilisè :

String	Description

\% Remplace le caractére littèral %
_ Remplace le caractére littèral _

```
mysql> SELECT 'David!' LIKE 'David\_';
        -> 0
mysql> SELECT 'David_' LIKE 'David\_';
        -> 1
```

Pour spècifier un caractére d'èchappement diffèrent, utilisez la clause `ESCAPE` :

```
mysql> SELECT 'David_' LIKE 'David|_' ESCAPE '|';
        -> 1
```

Les deux exemples suivants illustrent le fait que les comparaisons de chaînes de caractéres ne sont pas sensibles á la casse á moins qu'une des opèrandes soit une chaîne binaire.

```
mysql> SELECT 'abc' LIKE 'ABC';
        -> 1
mysql> SELECT 'abc' LIKE BINARY 'ABC';
        -> 0
```

`LIKE` est ègalement autorisè pour les expressions numèriques. (C'est une extension MySQL á la norme ANSI SQL `LIKE`.)

```
mysql> SELECT 10 LIKE '1%';
        -> 1
```

Note : Comme MySQL utilise la syntaxe d'èchappement de caractéres du langage C dans les chaînes (par exemple, '`\n`'), vous devez doubler tous les slashs '\' que vous utilisez dans les expressions `LIKE`. Par exemple, pour rechercher les nouvelles lignes ('`\n`'), vous devez le spècifier comme cela : '`\\n`'. Pour rechercher un anti-slash ('\'), vous devez le spècifier comme cela : '`\\\\`' (les anti-slashs sont supprimès une premiére fois pas l'analyseur syntaxique, puis une deuxiéme fois par le moteur d'expression règuliéres, ce qui ne laisse qu'un seul anti-slash á la fin).

`expr NOT LIKE pat [ESCAPE 'escape-char']`
Equivalent á `NOT (expr LIKE pat [ESCAPE 'escape-char'])`.

`expr REGEXP pat`
`expr RLIKE pat`
Effectue une recherche de chaîne avec l'expression règuliére `pat`. Le masque peut être une expression règuliére ètendue. Voir la section Voir Annexe G [Regexp], page 829. Retourne `1` si `expr` correspond au masque `pat`, sinon, retourne `0`. `RLIKE` est un synonyme de `REGEXP`, fourni pour assurer la compatiblitè avec `mSQL`. Note : Comme MySQL utilise la syntaxe d'èchappement de caractéres du langage C dans les chaînes (par exemple, '`\n`'), vous devez doubler tous les anti-slashs '\' que vous utilisez dans les expressions `REGEXP`. A partir de la version 3.23.4 de MySQL, `REGEXP` est insensible á la casse pour les comparaisons de chaînes normales (non binaires) :

```
mysql> SELECT 'Monty!' REGEXP 'm%y%%';
        -> 0
mysql> SELECT 'Monty!' REGEXP '.*';
```

```
        -> 1
mysql> SELECT 'new*\n*line' REGEXP 'new\\*.\\*line';
        -> 1
mysql> SELECT "a" REGEXP "A", "a" REGEXP BINARY "A";
        -> 1  0
mysql> SELECT "a" REGEXP "^[a-d]";
        -> 1
```

`REGEXP` et `RLIKE` utilise le jeu de caractéres courant (ISO-8859-1 Latin1 par dèfaut) lorsqu'il faut dècider du type d'un caractére.

`expr NOT REGEXP pat`
`expr NOT RLIKE pat`
: Identique á `NOT (expr REGEXP pat)`.

`STRCMP(expr1,expr2)`
: `STRCMP()` retourne `0` si les chaînes sont identiques, `-1` si la premiére chaîne est plus petite que la seconde et `1` dans les autres cas :

```
mysql> SELECT STRCMP('text', 'text2');
        -> -1
mysql> SELECT STRCMP('text2', 'text');
        -> 1
mysql> SELECT STRCMP('text', 'text');
        -> 0
```

`MATCH (col1,col2,...) AGAINST (expr)`
`MATCH (col1,col2,...) AGAINST (expr IN BOOLEAN MODE)`
: `MATCH ... AGAINST()` est utilisè pour les recherches en texte plein et retourne une information de pertinence - pertinence mesurè entre le texte des colonnes `(col1,col2,...)` et la valeur `expr`. La pertinence est un nombre á virgule flottante positif. La pertinence zèro signifie qu'il n'y a aucune similitude. `MATCH ... AGAINST()` est utilisable dans les versions 3.23.23 ou suivantes de MySQL. L'extension `IN BOOLEAN MODE` a ètè ajoutèe dans la version 4.0.1. Pour plus des dètails ainsi que des exemples, voir Section 6.8 [Fulltext Search], page 522.

6.3.2.2 Sensibilitè á la casse

`BINARY`
: L'opèrateur `BINARY` modifie la chaîne qui le suit en une chaîne binaire. C'est une solution simple pour forcer la comparaison de colonnes á être sensible á la casse même si la colonne n'est pas dèfinie comme ètant de type `BINARY` ou `BLOB` :

```
mysql> SELECT "a" = "A";
        -> 1
mysql> SELECT BINARY "a" = "A";
        -> 0
```

`BINARY string` est un raccourci pour `CAST(string AS BINARY)`. Voir Section 6.3.5 [Cast Functions], page 470. `BINARY` a ètè introduit dans MySQL á partir de la version 3.23.0.

Notez que dans quelques cas, MySQL n'est pas capable d'utiliser l'index efficacement lorsque vous modifiez une colonne indexèe en `BINARY`.

Si vous voulez comparer un champ de type `BLOB` d'une maniére insensible á la casse, vous pouvez toujours le convertir en majuscules avant d'effectuer la comparaison :

```
SELECT 'A' LIKE UPPER(blob_col) FROM nom_de_table;
```

Nous avons planifiè d'introduire bientôt de possibles modifications entre les diffèrents jeux de caractéres pour rendre les comparaisons de chaînes encore plus flexibles.

6.3.3 Fonctions numèriques

6.3.3.1 Opèrations arithmètiques

Les opèrateurs arithmètiques usuels sont disponibles. Notez que dans le cas de '`-`', '`+`', and '`*`', le rèsultat est calculè avec en `BIGINT` avec une prècision de 64 bits si les deux arguments sont des entiers ! Si l'un des arguments est un entier non signè, et que l'autre argument est aussi un entier, le rèsultat sera un entier non signè. Voir Section 6.3.5 [Cast Functions], page 470.

`+` Addition :

```
mysql> SELECT 3+5;
        -> 8
```

`-` Soustraction :

```
mysql> SELECT 3-5;
        -> -2
```

`*` Multiplication :

```
mysql> SELECT 3*5;
        -> 15
mysql> SELECT 18014398509481984*18014398509481984.0;
        -> 324518553658426726783156020576256.0
mysql> SELECT 18014398509481984*18014398509481984;
        -> 0
```

Le rèsultat du dernier calcul est incorrect car le rèsultat de la multiplication des deux entiers á dèpassè la capacitè de calcul de `BIGINT` (64 bits).

`/` Division :

```
mysql> SELECT 3/5;
        -> 0.60
```

La division par zèro produit un rèsultat `NULL` :

```
mysql> SELECT 102/(1-1);
        -> NULL
```

Une division sera calculèe en `BIGINT` seulement si elle est effectuèe dans un contexte oú le rèsultat est transformè en entier.

6.3.3.2 Fonctions mathèmatiques

Toutes les fonctions mathèmatiques retournent `NULL` en cas d'erreur.

`-` Le signe moins. Change le signe de l'argument :

```
mysql> SELECT - 2;
        -> -2
```

Notez que si cet opèrateur est utilisè avec une valeur de type `BIGINT`, la valeur retournèe sera de type `BIGINT`! Cela signifie que vous devez èviter d'utiliser l'opèrateur `-` sur les valeurs de type entier infèrieures á `-2^63`!

`ABS(X)` Retourne la valeur absolue de `X` :

```
mysql> SELECT ABS(2);
        -> 2
mysql> SELECT ABS(-32);
        -> 32
```

Cette fonction est utilisable avec les valeurs issues des champs `BIGINT`.

`SIGN(X)` Retourne le signe de l'argument sous la forme `-1`, `0`, ou `1`, selon que `X` est nègatif, zèro, ou positif :

```
mysql> SELECT SIGN(-32);
        -> -1
mysql> SELECT SIGN(0);
        -> 0
mysql> SELECT SIGN(234);
        -> 1
```

`MOD(N,M)`
`%` Modulo (èquivalent de l'opèrateur `%` dans le langage C). Retourne le reste de la division de `N` par `M` :

```
mysql> SELECT MOD(234, 10);
        -> 4
mysql> SELECT 253 % 7;
        -> 1
mysql> SELECT MOD(29,9);
        -> 2
```

Cette fonction ne pose pas de problémes avec les `BIGINT`.

`FLOOR(X)` Retourne la valeur entiére infèrieure de `X` :

```
mysql> SELECT FLOOR(1.23);
        -> 1
mysql> SELECT FLOOR(-1.23);
        -> -2
```

Notez que la valeur retournèe sera de type `BIGINT`!

`CEILING(X)`
Retourne la valeur entiére supèrieure de `X` :

```
mysql> SELECT CEILING(1.23);
        -> 2
mysql> SELECT CEILING(-1.23);
        -> -1
```

Notez que la valeur retournèe sera de type `BIGINT`!

`ROUND(X)` Retourne l'entier le plus proche de `X` :

```
mysql> SELECT ROUND(-1.23);
        -> -1
mysql> SELECT ROUND(-1.58);
        -> -2
mysql> SELECT ROUND(1.58);
        -> 2
```

Notez que le comportement de l'opèrateur `ROUND()`, lorsque l'argument est exactement entre deux entiers, dèpend de la librairie C active. Certaines arrondissent toujours á l'entier pair le plus proche, toujours vers le haut, toujours vers le bas, ou toujours vers zèro. Si vous avez besoin d'un certain type d'arrondissement, vous devez utiliser une fonction bien dèfinie comme `TRUNCATE()` ou `FLOOR()`.

`ROUND(X,D)`

Retourne l'argument `X`, arrondi á un nombre á `D` dècimales. Si `D` vaut `0`, le rèsultat n'aura ni de partie dècimale, ni de sèparateur de dècimal :

```
mysql> SELECT ROUND(1.298, 1);
        -> 1.3
mysql> SELECT ROUND(1.298, 0);
        -> 1
```

`EXP(X)` Retourne la valeur de `e` (la base des logarithmes naturels) èlevè á la puissance `X` :

```
mysql> SELECT EXP(2);
        -> 7.389056
mysql> SELECT EXP(-2);
        -> 0.135335
```

`LN(X)` Retourne le logarithme naturel de `X` (nèpèrien) :

```
mysql> SELECT LN(2);
        -> 0.693147
mysql> SELECT LN(-2);
        -> NULL
```

Cette fonction a ètè ajoutèe á MySQL á partir de la version 4.0.3. C'est un synonyme de la fonction `LOG(X)`.

`LOG(X)`
`LOG(B,X)` Appelèe avec un seul paramétre, cette fonction retourne le logarithme naturel (nèpèrien) de `X` :

```
mysql> SELECT LOG(2);
        -> 0.693147
```

```
mysql> SELECT LOG(-2);
        -> NULL
```

Appelèe avec deux paramétres, cette fonction retourne le logarithme naturel de `X` pour une base `B` arbitraire :

```
mysql> SELECT LOG(2,65536);
        -> 16.000000
mysql> SELECT LOG(1,100);
        -> NULL
```

Cette base arbitraire a ètè ajoutèe á MySQL á partir de la version 4.0.3. `LOG(B,X)` est l'èquivalent de `LOG(X)/LOG(B)`.

`LOG2(X)` Retourne le logarithme en base 2 de `X` :

```
mysql> SELECT LOG2(65536);
        -> 16.000000
mysql> SELECT LOG2(-100);
        -> NULL
```

`LOG2()` est utile pour trouver combien de bits sont nècessaires pour stocker un nombre. Cette fonction a ètè ajoutèe á MySQL á partir de la version 4.0.3. Dans les versions antèrieures, vous pouvez utiliser `LOG(X)/LOG(2)` en remplacement.

`LOG10(X)` Retourne le logarithme en base 10 de `X` :

```
mysql> SELECT LOG10(2);
        -> 0.301030
mysql> SELECT LOG10(100);
        -> 2.000000
mysql> SELECT LOG10(-100);
        -> NULL
```

`POW(X,Y)`
`POWER(X,Y)`

Retourne la valeur de `X` èlevèe á la puissance `Y` :

```
mysql> SELECT POW(2,2);
        -> 4.000000
mysql> SELECT POW(2,-2);
        -> 0.250000
```

`SQRT(X)` Retourne la racine carrèe de `X` :

```
mysql> SELECT SQRT(4);
        -> 2.000000
mysql> SELECT SQRT(20);
        -> 4.472136
```

`PI()` Retourne la valeur de PI. Par dèfaut, 5 dècimales sont retournèes, mais MySQL utilise la double prècision pour PI.

```
mysql> SELECT PI();
        -> 3.141593
mysql> SELECT PI()+0.000000000000000000;
        -> 3.141592653589793116
```

`COS(X)` Retourne le cosinus de `X`, oú `X` est donnè en radians :

```
mysql> SELECT COS(PI());
        -> -1.000000
```

`SIN(X)` Retourne le sinus de `X`, oú `X` est donnè en radians :

```
mysql> SELECT SIN(PI());
        -> 0.000000
```

`TAN(X)` Retourne la tangente de `X`, oú `X` est donnè en radians :

```
mysql> SELECT TAN(PI()+1);
        -> 1.557408
```

`ACOS(X)` Retourne l'arccosinus de `X`, c'est á dire, la valeur de l'angle dont `X` est la cosinus. Retourne `NULL` si `X` n'est pas dans l'intervalle `-1 - 1` :

```
mysql> SELECT ACOS(1);
        -> 0.000000
mysql> SELECT ACOS(1.0001);
        -> NULL
mysql> SELECT ACOS(0);
        -> 1.570796
```

`ASIN(X)` Retourne l'arcsinus de `X`, c'est á dire, la valeur de l'angle dont le sinus est `X`. Retourne `NULL` si `X` n'est pas dans l'intervalle `-1 - 1` :

```
mysql> SELECT ASIN(0.2);
        -> 0.201358
mysql> SELECT ASIN('foo');
        -> 0.000000
```

`ATAN(X)` Retourne l'arctangente de `X`, c'est á dire, la valeur de l'angle dont la tangente est `X` :

```
mysql> SELECT ATAN(2);
        -> 1.107149
mysql> SELECT ATAN(-2);
        -> -1.107149
```

`ATAN(Y,X)`
`ATAN2(Y,X)`

Retourne l'arctangente des variables `X` et `Y`. Cela revient á calculer l'arctangennte de `Y / X`, exceptè que les signes des deux arguments servent á dèterminer le quadrant du rèsultat :

```
mysql> SELECT ATAN(-2,2);
        -> -0.785398
mysql> SELECT ATAN2(PI(),0);
        -> 1.570796
```

`COT(X)` Retourne la cotangente de `X` :

```
mysql> SELECT COT(12);
        -> -1.57267341
mysql> SELECT COT(0);
        -> NULL
```

`RAND()`
`RAND(N)` Retourne un nombre alèatoire á virgule flottante comprit dans l'intervalle `0` - `1.0`. Si l'argument entier `N` est spècifiè, il est utilisè comme initilisation du gènèrateur de nombres alèatoires :

```
mysql> SELECT RAND();
        -> 0.9233482386203
mysql> SELECT RAND(20);
        -> 0.15888261251047
mysql> SELECT RAND(20);
        -> 0.15888261251047
mysql> SELECT RAND();
        -> 0.63553050033332
mysql> SELECT RAND();
        -> 0.70100469486881
```

Vous ne pouvez pas utiliser une colonne de valeur `RAND()` dans une clause `ORDER BY`, parce que `ORDER BY` va èvaluer la colonne plusieurs fois. Dans la version 3.23 de MySQL, vous pouvez, tout de même, faire ceci : `SELECT * FROM nom_de_table ORDER BY RAND()`

Cette syntaxe est trés pratique pour faire une sèlection alèatoire de lignes : `SELECT * FROM table1,table2 WHERE a=b AND c<d ORDER BY RAND() LIMIT 1000`.

Notez que la fonction `RAND()` dans une clause `WHERE` sera rèèvaluèe á chaque fois que `WHERE` sera exècutè.

`RAND()` n'est pas un gènèrateur de nombres alèatoires parfait, mais reste une maniére rapide de produire des nombres alèatoires ads-hoc portables selon les diffèrentes plate-formes pour une même version de MySQL.

`LEAST(X,Y,...)`
Avec deux arguments ou plus, cette fonction retourne la plus petite des valeurs des diffèrents arguments. Les arguments sont comparès en respectant les régles suivantes :

- Si la valeur retournèe est dans un contexte `INTEGER`, ou bien que tous les arguments sont des entiers, ils sont comparès en tant qu'entiers.
- Si la valeur retournèe est dans un contexte `REAL`, ou bien que tous les arguments sont des rèels, ils sont comparès en tant que rèels.
- Si l'un des arguments est une chaîne sensible á la casse, les arguments sont comparès comme des chaînes sensibles á la casse.
- Dans tous les autres cas, les arguments sont comparès comme des chaînes, insensibles á la casse.

```
mysql> SELECT LEAST(2,0);
        -> 0
mysql> SELECT LEAST(34.0,3.0,5.0,767.0);
        -> 3.0
mysql> SELECT LEAST("B","A","C");
        -> "A"
```

Dans les versions de MySQL 3.22.5 et antèrieures, vous pouvez utiliser `MIN()` á la place de `LEAST`.

`GREATEST(X,Y,...)`

Retourne le plus grand des arguments. Les arguments sont comparès en suivant les mêmes régles que la fonction `LEAST` :

```
mysql> SELECT GREATEST(2,0);
        -> 2
mysql> SELECT GREATEST(34.0,3.0,5.0,767.0);
        -> 767.0
mysql> SELECT GREATEST("B","A","C");
        -> "C"
```

Dans les versions de MySQL 3.22.5 et antèrieures, vous pouvez utiliser `MAX()` á la place de `GREATEST`.

`DEGREES(X)`

Retourne l'argument `X`, convertit de radians en degrès :

```
mysql> SELECT DEGREES(PI());
        -> 180.000000
```

`RADIANS(X)`

Retourne l'argument `X`, convertit de degrès en radians :

```
mysql> SELECT RADIANS(90);
        -> 1.570796
```

`TRUNCATE(X,D)`

Retourne l'argument `X`, tronquè á `D` dècimales. Si `D` vaut `0`, le rèsultat n'aura ni sèparateur dècimal, ni partie dècimale :

```
mysql> SELECT TRUNCATE(1.223,1);
        -> 1.2
mysql> SELECT TRUNCATE(1.999,1);
        -> 1.9
mysql> SELECT TRUNCATE(1.999,0);
        -> 1
mysql> SELECT TRUNCATE(-1.999,1);
        -> -1.9
```

A partir de MySQL 3.23.51 tous les nombres sont arrondis vers zèro.

Si `D` est nègatif, la partie entiére du nombre est mise á zèro :

```
mysql> SELECT TRUNCATE(122,-2);
       -> 100
```

Notez que les nombres dècimaux ne sont pas sotckès exactement comme les nombres entiers , mais comme des valeurs doubles. Vous pouvez être dupès par le rèsultat suivant :

```
mysql> SELECT TRUNCATE(10.28*100,0);
       -> 1027
```

Ce rèsultat est normal car 10.28 est actuellement stockè comme cela 10.2799999999999999.

6.3.4 Fonctions de dates et d'heures

Voir Section 6.2.2 [Date and time types], page 423 pour une description dètaillèe des intervalles de validitè de chaque type, ainsi que les formats valides de spècifications des dates et heures.

Voici un exemple d'utilisation des fonctions de date. La requête suivante sèlectionne toutes les lignes dont la colonne `date_col` reprèsente une date de moins de 30 jours :

```
mysql> SELECT quelquechose FROM nom_de_table
           WHERE TO_DAYS(NOW()) - TO_DAYS(date_col) <= 30;
```

`DAYOFWEEK(date)`
: Retourne l'index du jour de la semaine : pour `date` (`1` = Dimanche, `2` = Lundi, ... `7` = Samedi). Ces index correspondent au standart ODBC :

```
mysql> SELECT DAYOFWEEK('1998-02-03');
        -> 3
```

`WEEKDAY(date)`
: Retourne l'index du jour de la semaine, avec la convertion suivante : `date` (`0` = Lundi, `1` = Mardi, ... `6` = Dimanche) :

```
mysql> SELECT WEEKDAY('1997-10-04 22:23:00');
        -> 5
mysql> SELECT WEEKDAY('1997-11-05');
        -> 2
```

`DAYOFMONTH(date)`
: Retourne le jour de la date `date`, dans un intervalle de `1` á `31` :

```
mysql> SELECT DAYOFMONTH('1998-02-03');
        -> 3
```

`DAYOFYEAR(date)`
: Retourne le jour de la date `date`, dans un intervalle de `1` á `366` :

```
mysql> SELECT DAYOFYEAR('1998-02-03');
        -> 34
```

`MONTH(date)`
: Retourne le numèro du mois de la date `date`, dans un intervalle de `1` á `12` :

```
mysql> SELECT MONTH('1998-02-03');
        -> 2
```

`DAYNAME(date)`
: Retourne le nom du jour de la semaine, en anglais, de la date `date` :

```
mysql> SELECT DAYNAME("1998-02-05");
        -> 'Thursday'
```

`MONTHNAME(date)`
: Retourne le nom du mois de la date `date` :

```
mysql> SELECT MONTHNAME("1998-02-05");
        -> 'February'
```

`QUARTER(date)`

Retourne le numèro du trimestre de la date `date`, dans un intervalle de `1` á `4` :

```
mysql> SELECT QUARTER('98-04-01');
        -> 2
```

`WEEK(date)`
`WEEK(date,first)`

Avec un seul argument, retourne le numèro de la semaine dans l'année de la date `date`, dans un intervalle de `0` á `53` (oui, il peut y avoir un dèbut de semaine numèro 53), en considèrant que Dimanche est le premier jour de la semaine. Avec deux arguments, la fonction `WEEK()` vous permet de spècifier si les semaines commencent le Dimanche ou le Lundi et la valeur retournèe sera dans l'intervalle `0-53` ou bien `1-52`.

Voici un tableau explicatif sur le fonctionnement du second argument :

Value	`Meaning`
0	La semaine comme le Dimanche et retourne une valeur dans l'intervalle 0-53
1	La semaine comme le Lundi et retourne une valeur dans l'intervalle 0-53
2	La semaine comme le Dimanche et retourne une valeur dans l'intervalle 1-53
3	La semaine comme le Lundi et retourne une valeur dans l'intervalle 1-53 (ISO 8601)

```
mysql> SELECT WEEK('1998-02-20');
        -> 7
mysql> SELECT WEEK('1998-02-20',0);
        -> 7
mysql> SELECT WEEK('1998-02-20',1);
        -> 8
mysql> SELECT WEEK('1998-12-31',1);
        -> 53
```

Note : Dans la version 4.0 de MySQL, `WEEK(#,0)` a ètè modifiè pour être utilisè avec le calendrier USA.

Notez que si une semaine est la derniére semaine de l'année prècèdente, MySQL retournera 0 si vous n'utilisez pas 2 ou 3 comme argument optionnel :

```
mysql> SELECT YEAR('2000-01-01'), WEEK('2000-01-01',0);
        -> 2000, 0
mysql> SELECT WEEK('2000-01-01',2);
        -> 52
```

On peut prètendre que la fonction `WEEK()` de MySQL retourne `52`, vu que la date donnèe est en fait, la 52 éme semaine de l'année 1999.

Nous avons dècidè de retourner 0 au lieu de vouloir que la fonction retourne 'le numèro de la semaine dans une annèe donnèe'. Ceci rend l'utilisation de la fonction `WEEK()` fiable une fois combinèe avec d'autres fonctions qui extraient une partie de date á partir d'une date :

```
mysql> SELECT YEARWEEK('2000-01-01');
        -> 199952
mysql> SELECT MID(YEARWEEK('2000-01-01'),5,2);
        -> 52
```

`YEAR(date)`

Retourne l'année de la date `date`, dans un intervalle de `1000` á `9999`:

```
mysql> SELECT YEAR('98-02-03');
        -> 1998
```

`YEARWEEK(date)`
`YEARWEEK(date,first)`

Retourne l'année et la semaine pour une date. Le second argument fonctionne exactement comme le second argument de la fonction `WEEK()`. Notez que l'année peut être diffèrente de l'année de la date pouur la premiére et la derniére semaine de l'année :

```
mysql> SELECT YEARWEEK('1987-01-01');
        -> 198653
```

Notez que le numèro de la semaine est diffèrent de la valeur retournèe par la fonction `WEEK()` (`0`) pour un argument optionel 0 ou 1, alors que la fonction `WEEK()` retourne une semaine dans une annèee donnèee.

`HOUR(time)`

Retourne le nombre d'heures pour l'heure `time`, dans un intervalle de `0` á `23` :

```
mysql> SELECT HOUR('10:05:03');
        -> 10
```

`MINUTE(time)`

Retourne le nombre de minutes pour l'heure for `time`, dans un intervalle de `0` á `59` :

```
mysql> SELECT MINUTE('98-02-03 10:05:03');
        -> 5
```

`SECOND(time)`

Retourne le nombre de secondes pour l'heure `time`, dans un intervalle de `0` á `59` :

```
mysql> SELECT SECOND('10:05:03');
        -> 3
```

`PERIOD_ADD(P,N)`

Ajoute `N` mois á la pèriode `P` (au format `YYMM` ou `YYYYMM`). Retourne une valeur dans le format `YYYYMM`.

Notez que l'argument `P` **n'est pas** de type date :

```
mysql> SELECT PERIOD_ADD(9801,2);
        -> 199803
```

`PERIOD_DIFF(P1,P2)`

Retourne le nombre de mois entre les pèriodes `P1` et `P2`. `P1` et `P2` doivent être dans au format `YYMM` ou `YYYYMM`.

Notez que les arguments `P1` et `P2` **ne sont pas** de type date :

```
mysql> SELECT PERIOD_DIFF(9802,199703);
        -> 11
```

`DATE_ADD(date,INTERVAL expr type)`
`DATE_SUB(date,INTERVAL expr type)`
`ADDDATE(date,INTERVAL expr type)`
`SUBDATE(date,INTERVAL expr type)`

Ces fonctions effectuent des calculs sur les dates. Elles sont nouvelles depuis la version 3.22 de MySQL. `ADDDATE()` et `SUBDATE()` sont synonymes de `DATE_ADD()` and `DATE_SUB()`. Dans la version 3.23 de MySQL, vous pouvez utiliser les opèrateurs `+` et `-` en lieu et place des fonctions `DATE_ADD()` et `DATE_SUB()` si l'expression de droite est une colonne de type date ou datetime. (Voir l'exemple suivant) `date` est une valeur de type `DATETIME` ou `DATE`, indiquant la date du dèbut. `expr` est une expression spècifiant la valeur de l'intervalle á ajouter ou á soustraire depuis le dèbut de la date. `expr` est une chaîne ; elle peut commencer par le signe '`-`' pour les intervalles nègatifs. `type` est un mot clè indiquant comment l'expression doit être interprètèe.

La fonction `EXTRACT(type FROM date)` retourne le 'type' d'intervalle pour la date.

La table suivante illustre la relation entre les arguments `type` et `expr` :

`type` **value**	**Expected** `expr` **format**
SECOND	SECONDES
MINUTE	MINUTES
HOUR	HEURES
DAY	JOURS
MONTH	MOIS
YEAR	ANNES
MINUTE_SECOND	"MINUTES:SECONDES"
HOUR_MINUTE	"HEURES:MINUTES"
DAY_HOUR	"JOURS HEURES"
YEAR_MONTH	"ANNEES-MOIS"
HOUR_SECOND	"HEURES:MINUTES:SECONDES"
DAY_MINUTE	"JOURS HEURES:MINUTES"
DAY_SECOND	"JOURS HEURES:MINUTES:SECONDES"

MySQL autorise n'importe quel signe de ponctuation comme dèlimiteur dans `expr`. Ceux qui sont prèsentès dans la table ci-dessus ne sont que des suggestions. Si l'argument `date` est une `DATE` et que votre calcul implique seulement les annèes (`YEAR`), les mois (`MONTH`), et les jours (`DAY`) (c'est-á-dire, pas d'heures), le rèsultat est une `DATE`. Sinon, le rèsultat est une valeur de type `DATETIME` :

```
mysql> SELECT "1997-12-31 23:59:59" + INTERVAL 1 SECOND;
        -> 1998-01-01 00:00:00
mysql> SELECT INTERVAL 1 DAY + "1997-12-31";
        -> 1998-01-01
mysql> SELECT "1998-01-01" - INTERVAL 1 SECOND;
       -> 1997-12-31 23:59:59
mysql> SELECT DATE_ADD("1997-12-31 23:59:59",
```

```
    ->                 INTERVAL 1 SECOND);
        -> 1998-01-01 00:00:00
mysql> SELECT DATE_ADD("1997-12-31 23:59:59",
    ->                 INTERVAL 1 DAY);
        -> 1998-01-01 23:59:59
mysql> SELECT DATE_ADD("1997-12-31 23:59:59",
    ->                 INTERVAL "1:1" MINUTE_SECOND);
        -> 1998-01-01 00:01:00
mysql> SELECT DATE_SUB("1998-01-01 00:00:00",
    ->                 INTERVAL "1 1:1:1" DAY_SECOND);
        -> 1997-12-30 22:58:59
mysql> SELECT DATE_ADD("1998-01-01 00:00:00",
    ->                 INTERVAL "-1 10" DAY_HOUR);
        -> 1997-12-30 14:00:00
mysql> SELECT DATE_SUB("1998-01-02", INTERVAL 31 DAY);
        -> 1997-12-02
```

Si vous spècifiez un intervalle de valeur qui est trop court (c'est-á-dire qu'il n'inclut pas toutes les valeurs auxquelles ont peu s'attendre pour le type `type`), MySQL suppose que vous avez ignorè ces valeurs. Par exemple, si vous spècifiez un `type` de `DAY_SECOND` , la valeur `expr` doit avoir des jours, heures, minutes, secondes. Si vous spècifiez une valeur telle que `"1:10"`, MySQL supposera que les jours et les heures sont ignorès, et que la valeur fournit reprèsente uniquement les minutes et les secondes. En d'autres termes, `"1:10" DAY_SECOND` s'interpréte comme `"1:10" MINUTE_SECOND`. C'est comparable á la maniére dont MySQL interpréte les valeurs `TIME` qui reprèsente une durèe plutôt qu'une heure du jour.

Notez que si vous ajoutez ou soustrayez á une date une valeur qui contient des heures, le rèsultat final sera automatiquement converti en DATETIME :

```
mysql> SELECT DATE_ADD("1999-01-01", INTERVAL 1 DAY);
       -> 1999-01-02
mysql> SELECT DATE_ADD("1999-01-01", INTERVAL 1 HOUR);
       -> 1999-01-01 01:00:00
```

Si vous utilisez des dates incorrectes, le rèsultat sera `NULL`. Si vous ajoutez des `MONTH`, `YEAR_MONTH`, ou `YEAR`, et que le rèsultat dèpasse le nombre de jour dansle mois, ce nombre de jour sera ramenè au maximun acceptable :

```
mysql> SELECT DATE_ADD('1998-01-30', INTERVAL 1 MONTH);
        -> 1998-02-28
```

Notez que dans l'exemple prècèdent, le mot `INTERVAL` et le type `type` ne sont pas sensible á la casse.

`EXTRACT(type FROM date)`

La fonction `EXTRACT()` utilise les mêmes types d'intervalles que la fonction `DATE_ADD()` ou la fonction `DATE_SUB()`, mais extrait des parties de date plutôt que des opèrations de date.

```
mysql> SELECT EXTRACT(YEAR FROM "1999-07-02");
       -> 1999
```

```
mysql> SELECT EXTRACT(YEAR_MONTH FROM "1999-07-02 01:02:03");
       -> 199907
mysql> SELECT EXTRACT(DAY_MINUTE FROM "1999-07-02 01:02:03");
       -> 20102
```

`TO_DAYS(date)`

Retourne le nombre de jours depuis la date 0 jusqu'á la date `date` :

```
mysql> SELECT TO_DAYS(950501);
        -> 728779
mysql> SELECT TO_DAYS('1997-10-07');
        -> 729669
```

`TO_DAYS()` n'est pas fait pour travailler avec des dates qui prècédent l'avénement du calendrier Grègorien (1582), car elle ne prend pas en compte les jours perdus lors du changement de calendrier.

`FROM_DAYS(N)`

Retourne la date correspondant au nombre de jours (`N`) depuis la date 0 :

```
mysql> SELECT FROM_DAYS(729669);
        -> '1997-10-07'
```

`FROM_DAYS()` n'est pas fait pour travailler avec des dates qui prècédent l'avénement du calendrier Grègorien (1582), car elle ne prend pas en compte les jours perdus lors du changement de calendrier.

`DATE_FORMAT(date,format)`

Formate la date `date` avec le format `format`. Les spècificateurs suivants peuvent être utilisè dans la chaîne `format` :

Specifier	Description
%M	Nom du mois (January..December)
%W	Nom du jour de la semaine (Sunday..Saturday)
%D	Jour du mois, avec un suffix anglais (1st, 2nd, 3rd, etc.)
%Y	Année, au format numèrique, sur 4 chiffres
%y	Année, au format numèrique, sur 2 chiffres
%X	Année, pour les semaines qui commencent le Dimanche, au format numèrique, sur 4 chiffres, utilisè avec '%V'
%x	Année, pour les semaines qui commencent le Lundi, au format numèrique, sur 4 chiffres, utilisè avec '%v'
%a	Nom du jour de la semaine, en abrègè et en anglais (Sun..Sat)
%d	Jour du mois, au format numerique (00..31)
%e	Jour du mois, au format numerique (0..31)
%m	Mois, au format numerique (01..12)
%c	Mois, au format numèrique (1..12)
%b	Nom du mois, en abrègè et en anglais (Jan..Dec)
%j	Jour de l'année (001..366)
%H	Heure (00..23)
%k	Heure (0..23)
%h	Heure (01..12)
%I	Heure (01..12)

%l	Heure (1..12)
%i	Minutes, au format numerique (00..59)
%r	Heures, au format 12-heures (hh:mm:ss [AP]M)
%T	Heures, au format 24-heures (hh:mm:ss)
%S	Secondes (00..59)
%s	Secondes (00..59)
%p	AM ou PM
%w	Numèro du jour de la semaine (0=Sunday..6=Saturday)
%U	Numèro de la semaine (00..53), oú Dimanche est le premier jour de la semaine
%u	Numèro de la semaine (00..53), oú Lundi est le premier jour de la semaine
%V	Numèro de la semaine (01..53), oú Dimanche est le premier jour de la semaine, utilisè avec '%X'
%v	Numèro de la semaine (01..53), oú Lundi est le premier jour de la semaine, utilisè avec '%x'
%%	Un signe pourcentage littèral '%'.

Tous les autres caractéres sont simplement copiès dans le rèsultat sans interprètation:

```
mysql> SELECT DATE_FORMAT('1997-10-04 22:23:00', '%W %M %Y');
        -> 'Saturday October 1997'
mysql> SELECT DATE_FORMAT('1997-10-04 22:23:00', '%H:%i:%s');
        -> '22:23:00'
mysql> SELECT DATE_FORMAT('1997-10-04 22:23:00',
                          '%D %y %a %d %m %b %j');
        -> '4th 97 Sat 04 10 Oct 277'
mysql> SELECT DATE_FORMAT('1997-10-04 22:23:00',
                          '%H %k %I %r %T %S %w');
        -> '22 22 10 10:23:00 PM 22:23:00 00 6'
mysql> SELECT DATE_FORMAT('1999-01-01', '%X %V');
        -> '1998 52'
```

A partir de MySQL version 3.23, le caractére '%' est nècessaire avant tous les caractéres de spècification de format. Dans les anciennes versions, il ètait optionnel.

`TIME_FORMAT(time,format)`

Cette fonction est utilisèe exactement comme la fonction `DATE_FORMAT()` ci-dessus, mais la chaîne `format` ne doit utiliser que des spècificateurs d'heures, qui gérent les heures, minutes et secondes. Les autres spècificateurs gènéreront la valeur `NULL` ou `0`.

`CURDATE()`
`CURRENT_DATE`

Retourne la date courante au format `'YYYY-MM-DD'` ou `YYYYMMDD`, suivant le contexte numèrique ou chaîne :

```
mysql> SELECT CURDATE();
        -> '1997-12-15'
mysql> SELECT CURDATE() + 0;
```

```
                    -> 19971215
```

`CURTIME()`
`CURRENT_TIME`
: Retourne l'heure courante au format `'HH:MM:SS'` ou `HHMMSS`, suivant le contexte numèrique ou chaîne :

```
mysql> SELECT CURTIME();
        -> '23:50:26'
mysql> SELECT CURTIME() + 0;
        -> 235026
```

`NOW()`
`SYSDATE()`
`CURRENT_TIMESTAMP`
: Retourne la date courante au format `'YYYY-MM-DD HH:MM:SS'` ou `YYYYMMDDHHMMSS`, suivant le contexte numèrique ou chaîne :

```
mysql> SELECT NOW();
        -> '1997-12-15 23:50:26'
mysql> SELECT NOW() + 0;
        -> 19971215235026
```

Notez que `NOW()` est èvaluè une seule fois par requête, au dèbut de l'exècution de celle-ci. Cela signifie que si il y a de multiples rèfèrences á `NOW()` dans une requête, la même date sera utilisè.

`UNIX_TIMESTAMP()`
`UNIX_TIMESTAMP(date)`
: Lorsqu'elle est appelè sans argument, cette fonction retourne un timestamp Unix (nombre de secondes depuis `'1970-01-01 00:00:00'` GMT). Si `UNIX_TIMESTAMP()` est appelè avec un argument `date` , elle retourne le timestamp correspondant á cette date. `date` peut être une chaîne de type `DATE`, `DATETIME`, `TIMESTAMP`, ou un nombre au format `YYMMDD` ou `YYYYMMDD`, en horaire local :

```
mysql> SELECT UNIX_TIMESTAMP();
        -> 882226357
mysql> SELECT UNIX_TIMESTAMP('1997-10-04 22:23:00');
        -> 875996580
```

Lorsque `UNIX_TIMESTAMP` est utilisè sur une colonne de type `TIMESTAMP`, la fonction reçoit directement la valeur, sans conversion explicite. Si vous donnez á `UNIX_TIMESTAMP()` une date hors de son intervalle de validitè, elle retourne 0.

Si vous voulez soustraire une colonne de type `UNIX_TIMESTAMP()`, vous devez sûrement vouloir un rèsultat de type entier signè. Voir Section 6.3.5 [Cast Functions], page 470.

`FROM_UNIXTIME(unix_timestamp)`
: Retourne une reprèsentation de l'argument `unix_timestamp` sous la forme `'YYYY-MM-DD HH:MM:SS'` ou `YYYYMMDDHHMMSS`, suivant si la fonction est utilisè dans un contexte numèrique ou de chaîne.

```
mysql> SELECT FROM_UNIXTIME(875996580);
        -> '1997-10-04 22:23:00'
mysql> SELECT FROM_UNIXTIME(875996580) + 0;
        -> 19971004222300
```

`FROM_UNIXTIME(unix_timestamp,format)`
Retourne une chaîne représentant le timestamp Unix, formatè en accord avec la chaîne `format`. `format` doit contenir les mêmes spècificateurs que ceux utilisès par la fonction `DATE_FORMAT()` :

```
mysql> SELECT FROM_UNIXTIME(UNIX_TIMESTAMP(),
                            '%Y %D %M %h:%i:%s %x');
        -> '1997 23rd December 03:43:30 1997'
```

`SEC_TO_TIME(seconds)`
Retourne l'argument `seconds`, convertit en heures, minutes et secondes au format `'HH:MM:SS'` ou `HHMMSS`, suivant le contexte numèrique ou chaîne :

```
mysql> SELECT SEC_TO_TIME(2378);
        -> '00:39:38'
mysql> SELECT SEC_TO_TIME(2378) + 0;
        -> 3938
```

`TIME_TO_SEC(time)`
Retourne l'argument `time`, convertit en secondes :

```
mysql> SELECT TIME_TO_SEC('22:23:00');
        -> 80580
mysql> SELECT TIME_TO_SEC('00:39:38');
        -> 2378
```

6.3.5 Fonctions de transtypage

La syntaxe de la fonction `CAST` est :

```
CAST(expression AS type)

ou

CONVERT(expression,type)
```

Oú type est l'un des suivants :

- `BINARY`
- `DATE`
- `DATETIME`
- `SIGNED {INTEGER}`
- `TIME`
- `UNSIGNED {INTEGER}`

`CAST()` est la syntaxe ANSI SQL99 et `CONVERT()` est la syntaxe ODBC.

La fonction de transtypage est trés pratique lorsque vous voulez crèer une colonne avec un type spècifique dans une requête `CREATE ... SELECT` :

```
CREATE TABLE nouvelle_table SELECT CAST('2000-01-01' AS DATE);
```

`CAST(chaîne AS BINARY` est l'èquivalent de `BINARY chaîne`.

Pour transformer une chaîne de caractéres en une valeur numèrique, vous ne devez rien faire de particulier ; juste utiliser la valeur de la chaîne en lieu et place de la valeur numèrique :

```
mysql> SELECT 1+'1';
       -> 2
```

Si vous utilisez un nombre dans un contexte de chaîne, le nombre sera automatiquement converti en une chaîne binaire.

mysql> SELECT concat("salut toi ",2); -> "salut toi 2"

MySQL supporte l'arithmètique avec les valeurs 64 bits signèes et non signèes. Si vous utilisez une opèration numèrique (comme le signe `+`) et qu'un des opèrandes est de type `unsigned integer`, alors, le rèsultat sera une valeur non signè. Vous pouvez corriger cela en utilisant les opèrateurs de transtypages `SIGNED` et `UNSIGNED`, qui transformeront l'opèration respectivement en un entier signè sur 64 bits et un entier non signè sur 64 bits.

```
mysql> SELECT CAST(1-2 AS UNSIGNED)
        -> 18446744073709551615
mysql> SELECT CAST(CAST(1-2 AS UNSIGNED) AS SIGNED);
        -> -1
```

Notez que si l'une ou l'autre opèration est une valeur á virgule flottante (Dans ce contexte, `DECIMAL()` est considèrè comme une valeur á virgule flottante) le rèsultat devrait être une valeur á virgule flottante et ne sera pas affectè par la régle ci-dessus.

```
mysql> SELECT CAST(1 AS UNSIGNED) -2.0
        -> -1.0
```

Si vous utilisez une chaîne dans une opèration arithmètique, elle sera converti en un nombre á virgule flottante.

Les fonctions `CAST()` et `CONVERT()` ont ètè ajoutèes dans la version 4.0.2 de MySQL.

Le rendu des valeurs non signèes a ètè modifiè dans la version 4.0 de MySQL pour pouvoir supporter correctement les valeurs de type `BIGINT`. Si vous voulez utiliser du code fonctionnant dans la version 4.0 et la version 3.23 de MySQL (dans ce cas, vous ne pouvez probablement pas utiliser les fonctions de transtypage), vous pouvez utiliser l'astuce suivante pour avoir un rèsultat signè lorsque vous soustrayez deux colonnes d'entier non signè :

```
SELECT (unsigned_column_1+0.0)-(unsigned_column_2+0.0);
```

L'idèe est que les colonnes sont convertis en un point mobile avant de faire la soustraction.

Si vous rencontrez un probléme avec les colonnes `UNSIGNED` dans vos anciennes applications MySQL lorsque vous effectuez le port sous la version 4.0 de MySQL , vous pouvez utiliser l'option `--sql-mode=NO_UNSIGNED_SUBTRACTION` lorsque vous lancez `mysqld`. Notez cependant qu'aussi longtemps que vous employez ceci, vous ne serez pas capable d'utiliser efficacement les colonnes de type `UNSIGNED BIGINT`.

6.3.6 Autres fonctions

6.3.6.1 Fonctions sur les bits

MySQL utilise l'arithmètique des BIGINT (64-bits) pour les opèrations sur les bits. Ces opèrateurs travaillent donc sur 64 bits.

`|` OU bit-á-bit (OR)

```
mysql> SELECT 29 | 15;
        -> 31
```

Le rèsultat est un entier de 64 bits non signè.

`&` ET bit-á-bit (AND)

```
mysql> SELECT 29 & 15;
        -> 13
```

Le rèsultat est un entier de 64 bits non signè.

`^` XOR bit-á-bit

```
mysql> SELECT 1 ^ 1;
        -> 0
mysql> SELECT 1 ^ 0;
        -> 1
mysql> SELECT 11 ^ 3;
        -> 8
```

Le rèsultat est un entier de 64 bits non signè.

`<<` Dècale les bits de l'entier (`BIGINT`) sur la gauche :

```
mysql> SELECT 1 << 2;
        -> 4
```

Le rèsultat est un entier de 64 bits non signè.

`>>` Dècale les bits de l'entier (`BIGINT`) sur la droite :

```
mysql> SELECT 4 >> 2;
        -> 1
```

Le rèsultat est un entier de 64 bits non signè.

`~` Inverse tous les bits :

```
mysql> SELECT 5 & ~1;
        -> 4
```

Le rèsultat est un entier de 64 bits non signè.

`BIT_COUNT(N)`
Retourne le nombre de bits non nuls de l'argument `N` :

```
mysql> SELECT BIT_COUNT(29);
        -> 4
```

6.3.6.2 Fonctions diverses

`DATABASE()`
Retourne le nom de la base de donnèes courante :

```
mysql> SELECT DATABASE();
        -> 'test'
```

Si aucune base de donnèes n'a ètè sèlèctionnèe, `DATABASE()` retourne une chaîne vide.

`USER()`
`SYSTEM_USER()`
`SESSION_USER()`

Retourne l'utilisateur MySQL courant :

```
mysql> SELECT USER();
        -> 'davida@localhost'
```

A partir de la version 3.22.11 de MySQL, cette fonction inclut le nom de l'hôte du client tout comme le nom de l'utilisateur. Vous pouvez extraire le nom de l'utilisateur comme ceci (fonctionne ègalement si le nom de l'hôte est prèsent) :

```
mysql> SELECT SUBSTRING_INDEX(USER(),"@",1);
        -> 'davida'
```

`PASSWORD(str)`

Calcule un mot de passe chiffrè á partir de la chaîne `str`. C'est cette fonction qui est utilisè pour encrypter les mots de passes MySQL pour être stockès dans une colonne de type `Password` de la table `user` :

```
mysql> SELECT PASSWORD('badpwd');
        -> '7f84554057dd964b'
```

Le chiffrage par `PASSWORD()` n'est pas rèversible.

`PASSWORD()` n'est pas un chiffrage comparable á la fonction Unix de chiffrage. Si votre mot de passe Unix et votre mot de passe MySQL sont identiques, il n'est ABSOLUMENT pas certain que Unix et MySQL le chiffreront de la même façon. Voir `ENCRYPT()`.

`ENCRYPT(str[,salt])`

Chiffre la chaîne `str` en utilisant la fonction `crypt()`. L'argument `salt` doit être une chaîne de deux caractéres. (A partir de la version 3.22.16, l'argument `salt` peut être plus long que deux caractéres.) :

```
mysql> SELECT ENCRYPT("hello");
        -> 'VxuFAJXVARROc'
```

Si la fonction `crypt()` n'est pas disponible sur votre systéme, la fonction `ENCRYPT()` retourenra toujours `NULL`.

La fonction `ENCRYPT()` conserve uniquement les 8 premiers caractéres de la chaîne `str`, au moins, sur certains systéme. Le comportement exact est directement dèterminè par la fonction systéme `crypt()` sous-jacente.

`ENCODE(str,pass_str)`

Chiffre la chaîne `str` en utilisant la clè `pass_str`. Pour dèchiffrer le rèsultat, utilisez la fonction `DECODE()`.

Le rèsultat est une chaîne binaire de la même longueur que `string`. Si vous voulez sauvegarder le rèsultat dans une colonne, utilisez une colonne de type `BLOB`.

`DECODE(crypt_str,pass_str)`
: Dèchiffre la chaîne chiffrèe `crypt_str` en utilisant la clè `pass_str`. `crypt_str` doit être une chaîne qui a ètè renvoyèe par la fonction `ENCODE()`.

`MD5(string)`
: Calcul la somme de vèrification MD5 de la chaîne `string`. La valeur retournèe est un entier hexadècimal de 32 caractéres qui peut être utilisè, par exemple, comme clè de hachage :

```
mysql> SELECT MD5("testing");
        -> 'ae2b1fca515949e5d54fb22b8ed95575'
```

C'est l'algorithme RSA ("RSA Data Security, Inc. MD5 Message-Digest Algorithm").

`SHA1(string)`
`SHA(string)`
: Calcul la somme de vèrification SHA1 160 bit de la chaîne `string`, comme dècrit dans la RFC 3174 (Secure Hash Algorithm). La valeur retournèe est un entier exadècimal de 40 caractéres, ou bien `NULL` dans le cas oú l'argument vaut `NULL`. Une des possibilitès d'utilisation de cette fonction est le hachage de clè. Vous pouvez aussi l'utilisè comme fonction de cryptographie sûre pour stocker les mots de passe.

```
mysql> SELECT SHA1("abc");
        -> 'a9993e364706816aba3e25717850c26c9cd0d89d'
```

La fonction `SHA1()` a ètè ajoutèe dans la version 4.0.2 de MySQL et peut être considèrèe comme une mèthode de cryptographie plus sûre que la fonction `MD5()`. La fonction `SHA()` est un alias de la fonction `SHA1()`.

`AES_ENCRYPT(string,key_string)`
`AES_DECRYPT(string,key_string)`
: Ces fonctions permettent le chiffrement/dèchiffrement de donnèes utilisant l'algorithme AES (Advanced Encryption Standard), anciennement connu sous le nom de Rijndael. Une clè de 128 bits est utilisè pour le chiffrement, mais vous pouvez l'ètendre á 256 bits en patchant la source. Nous avons choisi 128 bits parce que c'est plus rapide et assez sècurisè.

Les arguments peuvent être de n'importe quelle taille. Si l'un des arguments est `NULL`, le rèsultat de cette fonction sera `NULL`.

Vu que AES est un algorithme de niveau bloc, le capitonnage est utilisè pour chiffrer des chaînes de longueur inègales et donc, la longueur de la chaîne rèsultante peut être calculèe comme ceci : 16*(trunc(string_length/16)+1).

Si la fonction `AES_DECRYPT()` dètecte des donnèes invalides ou un capitonnage incorrect, elle retournera `NULL`. Il est ègalement possible que la fonction `AES_DECRYPT()` retourne une valeur diffèrente de `NULL` (valeur incohèrente) si l'entrèe de donnèes ou la clè est invalide.

Vous pouvez utiliser les fonctions AES pour stocker des donnèes sous une forme chiffrèe en modifiant vos requêtes:

```
INSERT INTO t VALUES (1,AES_ENCRYPT("text","password"));
```

Vous pouvez obtenir encore plus de sècuritè en èvitant de transfèrer la clè pour chaque requête, en la stockant dans une variable sur le serveur au moment de la connexion :

```
SELECT @password:="my password";
INSERT INTO t VALUES (1,AES_ENCRYPT("text",@password));
```

Les fonctions `AES_ENCRYPT()` et `AES_DECRYPT()` ont ètè ajoutèes dans la version 4.0.2 de MySQL et peuvent être considèrèes comme ètant les fonctions de cryptographie les plus sûres disponibles actuellement dans MySQL.

`DES_ENCRYPT(string_to_encrypt [, (key_number | key_string) ] )`

Chiffre la chaîne avec la clè donnèe en utilisant l'algorithme DES.

Notez que cette fonction fonctionne uniquement si vous avez configurè MySQL avec le support SSL. Voir Section 4.3.9 [Secure connections], page 239.

La clè de hachage utilisèe est choisie en suivant les recommandations suivantes :

Argument	**Description**
Un seul argument	La premiére clè de `des-key-file` est utilisèe.
Un numèro de clè	Le numèro de la clè donnèe (0-9) de `des-key-file` est utilisèe.
Une chaîne	La chaîne donnèe `key_string` doit être utilisè pour chiffrer `string_to_encrypt`.

La chaîne retournèe doit etre une chaîne binaire oú le premier caractére doit être `CHAR(128 | key_number)`.

Le nombre 128 a ètè ajoutè pour reconnaître facilement une clè de hachage. Si vous utilisez une chaîne comme clè, `key_number` doit être 127.

Si une erreur survient, la fonction retournera `NULL`.

La longueur de la chaîne de rèsultat doit être : `new_length= org_length + (8-(org_length % 8))+1`.

`des-key-file` a le format suivant :

```
key_number des_key_string
key_number des_key_string
```

Chaque `key_number` doit être un nombre dans l'intervalle 0 á 9. Les lignes dans le fichier peuvent être dans n'importe quel ordre. `des_key_string` est la chaîne qui permettera le chiffrage du message. Entre le nombre et la clè, il doit y avoir au moins un espace. La premiére clè est la clè par dèfaut qui sera utilisè si vous ne spècifiez pas d'autres clès en arguments de la fonction `DES_ENCRYPT()`.

Vous pouvez demander á MySQL de lire de nouvelles valeurs de clè dans le fichier de clès avec la commande `FLUSH DES_KEY_FILE`. Cela requiére le privilége `Reload_priv`.

Un des bènèfices d'avoir une liste de clès par dèfaut est que cela donne auux applications la possibilitè de regarder l'existence de la valeur chiffrèe de la colonne, sans pour autant donner la possibilitè á l'utilisateur final de dèchiffrer ces valeurs.

```
mysql> SELECT customer_address FROM customer_table WHERE
       crypted_credit_card = DES_ENCRYPT("credit_card_number");
```

`DES_DECRYPT(string_to_decrypt [, key_string])`

Dèchiffre une chaîne chiffrèe á l'aide de la fonction `DES_ENCRYPT()`.

Notez que cette fonction fonctionne uniquement si vous avez configurè MySQL avec le support SSL. Voir Section 4.3.9 [Secure connections], page 239.

Si l'argument `key_string` n'est pas donnè, la fonction `DES_DECRYPT()` examine le premier bit de la chaîne chiffrèe pour dèterminer le numèro de clè DES utilisè pour chiffrer la chaîne originale, alors la clè est lu dans le fichier `des-key-file` pour dèchiffrer le message. Pour pouvoir utiliser cela, l'utilisateur doit avoir le privilége `SUPER`.

Si vous passè l'argument `key_string` á cette fonction, cette chaîne est utilisèe comme clè pour dèchiffrer le message.

Si la chaîne `string_to_decrypt` ne semble pas être une chaîne chiffrèe, MySQL retournera la chaîne `string_to_decrypt`.

Si une erreur survient, cette fonction retourne `NULL`.

`LAST_INSERT_ID([expr])`

Retourne la derniére valeur gènèrèe automatiquement qui a ètè insèrèe dans une colonne de type `AUTO_INCREMENT`. Voir Section 8.4.3.30 [`mysql_insert_id()`], page 635.

```
mysql> SELECT LAST_INSERT_ID();
        -> 195
```

Le dernier ID gènèrè est conservè par le serveur pour chaque connexion. Un autre client ne le modifiera donc pas. Il ne sera pas modifiè non plus si vous modifiez directement la valeur d'une colonne `AUTO_INCREMENT` avec une valeur simple (c'est-á-dire, une valeur qui n'est ni NULL, ni 0).

Si vous insèrez plusieurs lignes au même moment avec une requête insert, `LAST_INSERT_ID()` retourne la valeur de la premiére ligne insèrèe. La raison de cela est de rendre possible de reproduire facilement la même requête `INSERT` sur d'autres serveurs.

Si `expr` est donnèe en argument á la fonction `LAST_INSERT_ID()`, alors la valeur de l'argument sera retournè par la fonction et sera enregistrè comme ètant la prochaine valeur retournèe par `LAST_INSERT_ID()`. Cela peut être utilisè pour simuler des sèquences :

Commencez par crèer la table suivante :

```
mysql> CREATE TABLE sequence (id INT NOT NULL);
mysql> INSERT INTO sequence VALUES (0);
```

Alors cette table peut être utilisè pour gènèrer des sèquences de nombres, comme ceci :

```
mysql> UPDATE sequence SET id=LAST_INSERT_ID(id+1);
```

Vous pouvez gènèrer des sèquences sans appeler la fonction `LAST_INSERT_ID()`, mais l'utilitè d'utiliser cette fonction cette fois si est que la valeur ID est gèrèe par le serveur comme ètant la derniére valeur gènèrèe automatiquement. (sècuritè multi-utilisateur). Vous pouvez retrouver la nouvelle ID tout comme vous pouvez lire n'importe quelle valeur `AUTO_INCREMENT` dans MySQL.

Par exemple, la fonction `LAST_INSERT_ID()` (sans argument) devrait retourner la nouvelle ID. La fonction C de l'API `mysql_insert_id()` peut être ègalement utilisèe pour trouver cette valeur.

Notez que la fonction `mysql_insert_id()` est incrèmentèe uniquement aprés des requêtes `INSERT` et `UPDATE`, donc, vous ne pouvez pas utiliser la fonction C de l'API pour trouver la valeur de `LAST_INSERT_ID(expr)` aprés avoir exècutè d'autres types de requêtes, comme `SELECT` ou bien `SET`.

`FORMAT(X,D)`

Formate l'argument `X` en un format comme `'#,###,###.##'`, arrondi á `D` dècimales. Si `D` vaut `0`, le rèsultat n'aura ni sèparateur dècimal, ni partie dècimale :

```
mysql> SELECT FORMAT(12332.123456, 4);
        -> '12,332.1235'
mysql> SELECT FORMAT(12332.1,4);
        -> '12,332.1000'
mysql> SELECT FORMAT(12332.2,0);
        -> '12,332'
```

`VERSION()`

Retourne une chaîne indiquant la version courante du serveur MySQL :

```
mysql> SELECT VERSION();
        -> '3.23.13-log'
```

Notez que si votre version se termine par `-log`, cela signifie que le systéme d'historique est actif.

`CONNECTION_ID()`

Retourne l'identifiant de connexion courant (`thread_id`). Chaque connexion a son propre identifiant unique :

```
mysql> SELECT CONNECTION_ID();
        -> 1
```

`GET_LOCK(str,timeout)`

Tente de poser un verrou nommè `str`, avec un dèlai d'expiration (`timeout`) exprimè en seconde. Retourne `1` si le verrou a ètè posè avec succés, `0` si il n'a pas pu être posè avant l'expiration du dèlai et `NULL` si une erreur est survenu (comme par exemple un manque de mèmoire, ou la mort du thread lui-même, par `mysqladmin kill`). Un verrou sera levè lorsque vous exècuterez la commande `RELEASE_LOCK()`, `GET_LOCK()` ou si le thread se termine. Cette fonction peut être utilisèe pour implèmenter des verrous applicatifs ou pour simuler des verrous de lignes. Les requêtes concurentes des autres clients de même nom seront bloquèes ; les clients qui s'entendent sur un nom de verrou peuvent les utiliser pour effectuer des verrouillages coopèratifs :

```
mysql> SELECT GET_LOCK("lock1",10);
        -> 1
mysql> SELECT IS_FREE_LOCK("lock2");
        -> 1
mysql> SELECT GET_LOCK("lock2",10);
```

```
        -> 1
mysql> SELECT RELEASE_LOCK("lock2");
        -> 1
mysql> SELECT RELEASE_LOCK("lock1");
        -> NULL
```

Notez que le deuxiéme appel á `RELEASE_LOCK()` retourne `NULL` car le verrou `"lock1"` a ètè automatiquement libèrè par le deuxiéme appel á `GET_LOCK()`.

`RELEASE_LOCK(str)`

Libére le verrou nommè `str`, obetnu par la fonction `GET_LOCK()`. Retourne `1` si le verrou a bien ètè libèrè, `0` si le verrou n'a pas ètè libèrè par le thread (dans ce cas, le verrou reste posè) et `NULL` si le nom du verrou n'existe pas. Le verrou n'existe pas si il n'a pas ètè obtenu par la fonction `GET_LOCK()` ou si il a dèjá ètè libèrè.

La commande `DO` est utilisable avec `RELEASE_LOCK()`. Voir Section 6.4.10 [DO], page 503.

`IS_FREE_LOCK(str)`

Regarde si le verrou nommè `str` peut être librement utilisè (i.e., non verrouillè). Retourne `1` si le verrou est libre (personne ne l'utilise), `0` si le verrou est actuellement utilisè et `NULL` si une erreur survient (comme un argument incorrect).

`BENCHMARK(count,expr)`

La fonction `BENCHMARK()` exècute l'expression `expr` de maniére rèpètèe `count` fois. Elle permet de tester la vèlocitè dde MySQL lors du traitement d'une requête. Le rèsultat est toujours `0`. L'objectif de cette fonction ne se voit que du côtè client, qui permet á ce dernier d'afficher la durée d'exècution de la requête :

```
mysql> SELECT BENCHMARK(1000000,ENCODE("bonjour","au revoir"));
+------------------------------------------------+
| BENCHMARK(1000000,ENCODE("bonjour","au revoir")) |
+------------------------------------------------+
|                                              0 |
+------------------------------------------------+
1 row in set (4.74 sec)
```

Le temps affichè est le temps côtè client, et non pas les ressources processeurs consommèes. il est conseillè d'utiliser `BENCHMARK()` plusieurs fois de suite pour interprèter un rèsultat, en dehors de charges ponctuelles sur le serveur.

`INET_NTOA(expr)`

Retourne l'adresse rèseau (4 ou 8 octets), de l'expression numèrique `exp` :

```
mysql> SELECT INET_NTOA(3520061480);
       ->  "209.207.224.40"
```

`INET_ATON(expr)`

Retourne un entier qui reprèsente l'expression numèrique de l'adresse rèseau. Les adresses peuvent être des entiers de 4 ou 8 octets.

```
mysql> SELECT INET_ATON("209.207.224.40");
```

```
    -> 3520061480
```

Le nombre gènèrè est toujours dans l'ordre des octets rèseau ; par exemple, le nombre prècèdent est caculè comme ceci : `209*256^3 + 207*256^2 + 224*256 +40`.

`MASTER_POS_WAIT(log_name, log_pos)`
: Bloque le maître jusqu'á ce que l'esclave atteigne une position donnèe dans le fichier d'historique principal, durant une rèplication. Si l'historique principal n'est pas initialisè, retourne `NULL`. Si l'esclave n'est pas dèmarrè, le maître restera bloquè jusqu'á ce que l'esclave soit dèmarrè et ai atteint la position demandèe. Si l'esclave a dèjá dèpassè cette position, la fonction se termine immèdiatement. La valeur retournèe est le nombre d'èvénements qui a du être traitè pour atteindre la position demandèe, ou `NULL` en cas d'erreur. Cette fonction est trés utile pour contrôler la synchronisation maître-esclave, mais elle a ètè initialement ècrite pour faciliter les tests de rèplications.

`FOUND_ROWS()`
: Retourne le nombre de lignes que la derniére commande `SELECT SQL_CALC_FOUND_ROWS ...` a retournè, si aucune limite n'a ètè dèfini par `LIMIT`.

```
mysql> SELECT SQL_CALC_FOUND_ROWS * FROM nom_de_table
       WHERE id > 100 LIMIT 10;
mysql> SELECT FOUND_ROWS();
```

Le second `SELECT` retourne un nombre indiquant combien de lignes le premier `SELECT` aurait retournè si il n'avait pas ètè ècrit avec une clause`LIMIT`.

Notez que si vous utilisez `SELECT SQL_CALC_FOUND_ROWS ...`, MySQL calcule toutes les lignes dans la liste des rèsultats. Ainsi, c'est plus rapide si vous n'utilisez pas de clause `LIMIT` et que la liste des rèsultats n'a pas besoin d'être envoyèe au client.

`SQL_CALC_FOUND_ROWS` est utilisable á partir de la version 4.0.0 de MySQL.

6.3.7 Fonctions avec la clause `GROUP BY`

Si vous utilisez une fonction de groupement avec une commande qui n'utilise pas la clause `GROUP BY`, cela revient á regrouper toutes les lignes ensembles.

`COUNT(expr)`
: Retourne le nombre de valeurs non-nulles (`NULL`) dans les lignes lues par la commande `SELECT` :

```
mysql> SELECT etudiant.nom_etudiant,COUNT(*)
    ->        FROM etudiant,course
    ->        WHERE etudiant.id_etudiant=course.id_etudiant
    ->        GROUP BY nom_etudiant;
```

`COUNT(*)` est lègérement diffèrent car il retourne le nombre de lignes lues, qu'elles contiennent ou pas la valeur `NULL`.

`COUNT(*)` est optimisè pour retourner trés rapidement le rèsultat si la commande `SELECT` n'opére que sur une table, qu'aucune autre valeur n'est calculèe en même temps, et qu'il n'y a pas de clause `WHERE`. Par exemple :

```
mysql> SELECT COUNT(*) FROM etudiant;
```

`COUNT(DISTINCT expr,[expr...])`
Retourne le nombre de lignes ayant des valeurs distinctes non-nulles (`NULL`) :

```
mysql> SELECT COUNT(DISTINCT resultat) FROM etudiant;
```

Avec MySQL, vos pouvez connaître le nombre de combinaison d'expressions distinctes qui ne contiennent pas la valeur `NULL` en donnant une liste d'expressions. En ANSI SQL, vous devriez faire la concatènation de toutes les expressions de clause `COUNT(DISTINCT ...)`.

`AVG(expr)`
Retourne la valeur moyenne de la colonne `expr` :

```
mysql> SELECT nom_etudiant, AVG(test_score)
    ->        FROM etudiant
    ->        GROUP BY nom_etudiant;
```

`MIN(expr)`
`MAX(expr)`
Retourne le minimum ou le maximum de l'expression `expr`. `MIN()` et `MAX()` acceptent aussi les chaînes comme argument ; dans ce cas, elles retournent le minimum ou le maximum de la valeur de la chaîne. Voir Section 5.4.3 [MySQL indexes], page 384.

```
mysql> SELECT nom_etudiant, MIN(test_score), MAX(test_score)
    ->        FROM etudiant
    ->        GROUP BY nom_etudiant;
```

Dans `MIN()`, `MAX()` et les autres fonctions d'agrègat, MySQL compare les colonnes de type `ENUM` et `SET` par la valeur de leur chaîne au lieu de la position relative de la chaîne dans la liste. Cela sera rectifiè.

`SUM(expr)`
Retourne la somme de l'expression `expr`. Notez que si aucune ligne n'est sèlectionnèe, la fonction retournera NULL !

`STD(expr)`
`STDDEV(expr)`
Retourne la dèviation standard de l'expression `expr`. C'est une extension á la norme ANSI SQL. La fonction `STDDEV()` est fournie pour assurer la compatiblitè avec Oracle.

`BIT_OR(expr)`
Retourne le `OR` bit-á-bit de l'expression `expr`. Les calculs sont effectuès avec une prècision de 64 bits (`BIGINT`).

`BIT_AND(expr)`
Retourne le `AND` bit-á-bit de l'expression `expr`. Les calculs sont effectuès avec une prècisioon de 64 bits (`BIGINT`).

MySQL a ètendu l'utilisation de la clause `GROUP BY`. Vous pouvez utiliser des noms de colonnes ou des expressions qui n'apparaissent pas dans la clause `GROUP BY`. Elles prennent alors *n'importe quelle valeur possible dans ce groupe*. Vous pouvez les utiliser pour amèliorer les performances, en èvitant de trier et regrouper des èlèments non critiques. Par exemple, vous n'avez pas besoin de faire de groupement par `client.nom` dans la requête suivante :

```
mysql> SELECT order.custid,client.nom,MAX(payments)
    ->        FROM order,client
    ->        WHERE order.custid = client.custid
    ->        GROUP BY order.custid;
```

En ANSI SQL, vous devez ajouter `client.nom` á la clause `GROUP BY`. En MySQL, le nom est redondant si vous n'utilisez par le mode ANSI. **N'utilisez surtout pas cette caractèristique** si les colonnes que vous omettez n'ont pas de valeur unique au sein du groupe! Vous obtiendriez des rèsultas incohèrents. Dans certains cas, vous pouvez utiliser `MIN()` et `MAX()` pour obtenir une valeur d'une colonne spècifique, même si elle n'est pas unique. La fonction suivante calcule la valeur de `column` dans la ligne contenant la plus petite valeur de la colonne `sort` :

```
SUBSTR(MIN(CONCAT(RPAD(sort,6,' '),column)),7)
```

Voir Section 3.5.4 [example-Maximum-column-group-row], page 180.

Notez que si vous utilisez la version 3.22 de MySQL (ou plus rècent), ou si vous essayez d'utiliser le mode ANSI SQL, vous ne pouvez pas utiliser d'expressions dans les clauses `GROUP BY` et `ORDER BY`. Vous pouvez contourner ces limitations en utilisant un alias d'expression :

```
mysql> SELECT id,FLOOR(value/100) AS val FROM nom_de_table
    ->        GROUP BY id,val ORDER BY val;
```

En MySQL version 3.23, vous pouvez faire :

```
mysql> SELECT id,FLOOR(value/100) FROM nom_de_table ORDER BY RAND();
```

6.4 Manipulation de donnèes : SELECT, INSERT, UPDATE, DELETE

6.4.1 Syntaxe de SELECT

```
SELECT [STRAIGHT_JOIN]
       [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
       [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS] [HIGH_PRIORITY]
       [DISTINCT | DISTINCTROW | ALL]
    select_expression,...
    [INTO {OUTFILE | DUMPFILE} 'nom_fichier' export_options]
    [FROM table_references
      [WHERE where_definition]
      [GROUP BY {unsigned_integer | nom_de_colonne | formula} [ASC | DESC], ...
      [HAVING where_definition]
      [ORDER BY {unsigned_integer | nom_de_colonne | formula} [ASC | DESC] ,...]
      [LIMIT [offset,] lignes]
      [PROCEDURE procedure_name(argument_list)]
      [FOR UPDATE | LOCK IN SHARE MODE]]
```

`SELECT` est utilisè pour obtenir des enregistrements venant d'une ou plusieurs tables. `select_expression` indique les champs que vous voulez obtenir. `SELECT` peut aussi être utilisè pour obtenir des enregistrements calculès sans aucune rèfèrence á une table. Par exemple :

```
mysql> SELECT 1 + 1;
         -> 2
```

Tous les mots-clès utilisès doivent être donnès exactement dans le même ordre que ci-dessus. Par exemple, une clause `HAVING` doit être placèe aprés toute clause `GROUP BY` et avant toute clause `ORDER BY`.

- Une expression `SELECT` peut être aliasèe en utilisant `AS`. L'alias est utilisè de la même façon que le nom du champ et peut être employè avec des clauses `ORDER BY` ou `HAVING`. Par exemple :

  ```
  mysql> SELECT CONCAT(last_name,', ',first_name) AS full_name
      FROM mytable ORDER BY full_name;
  ```

- Il n'est pas possible d'utiliser un alias de champ dans une clause `WHERE`, car la valeur du champ peut ne pas être dèfinie lorsque la clause `WHERE` est exècutèe. Voir Section A.5.4 [Problems with alias], page 705.

- La clause `FROM table_references` indique les tables á partir desquelles nous allons obtenir les enregistrements. Si vous indiquez le nom de plusieurs tables, vous faites une jointure. Pour davantage d'informations sur la syntaxe des jointures, consultez Section 6.4.1.1 [`JOIN`], page 486. Pour chaque table spècifièe, vous pouvez èventuellement indiquer un alias.

  ```
  table_name [[AS] alias] [USE INDEX (key_list)] [IGNORE INDEX (key_list)]
  ```

  Depuis la version 3.23.12 de MySQL, vous pouvez donner des indications sur les index que MySQL doit utiliser lors de l'obtention des enregistrements venant d'une table. C'est pratique lorsque `EXPLAIN` montre que MySQL utilise le mauvais index parmi la liste des index possibles. En indiquant `USE INDEX (key_list)`, vous pouvez dire á MySQL d'utiliser un seul des index possibles pour trouver des enregistrements dans la table. L'autre syntaxe `IGNORE INDEX (key_list)` peut être utilisèe pour dire á MySQL de ne pas utiliser un index en particulier.

  `USE/IGNORE KEY` sont des synonymes de `USE/IGNORE INDEX`.

- Vous pouvez vous rèfèrer á une table avec `nom_de_table` (au sein de la base de donnèes courante), ou avec `dbname.nom_de_table` pour expliciter le nom de la base de donnèes. Vous pouvez vous rèfèrer á un champ avec `nom_de_colonne`, `nom_de_table.nom_de_colonne`, ou `db_name.nom_de_table.nom_de_colonne`. Vous n'êtes pas obligès d'indiquer de prèfixe `nom_de_table` ou `db_name.nom_de_table` pour une rèfèrence á un champ dans un `SELECT`, á moins que la rèfèrence ne soit ambigüe. Consultez Section 6.1.2 [Legal names], page 407, pour des exemples d'ambigüitès qui nècessitent des formes plus explicites de rèfèrence á des champs.

- Une rèfèrence á une table peut être aliasèe en utilisant `nom_de_table [AS] alias_name` :

  ```
  mysql> SELECT t1.name, t2.salary FROM employee AS t1, info AS t2
      ->        WHERE t1.name = t2.name;
  mysql> SELECT t1.name, t2.salary FROM employee t1, info t2
      ->        WHERE t1.name = t2.name;
  ```

- Vous pouvez faire rèfèrence aux champs sèlectionnès en sortie dans des clauses `ORDER BY` et `GROUP BY` en utilisant les noms des champs, les alias des champs ou bien les positions des champs. Les positions des champs commencent á 1 :

```
mysql> SELECT college, region, seed FROM tournament
    ->        ORDER BY region, seed;
mysql> SELECT college, region AS r, seed AS s FROM tournament
    ->        ORDER BY r, s;
mysql> SELECT college, region, seed FROM tournament
    ->        ORDER BY 2, 3;
```

  Pour trier dans l'ordre inverse, ajoutez le mot-clè `DESC` (descendant) au nom du champ dans la clause `ORDER BY` qui vous permet de trier. Par dèfaut, l'ordre ascendant est utilisè; ceci peut être indiquè de façon explicite en utilisant le mot-clè `ASC`.
- Vous pouvez, dans une clause `WHERE`, utiliser n'importe laquelle des fonctions que MySQL supporte. Voir Section 6.3 [Functions], page 436.
- La clause `HAVING` peut faire rèfèrence á n'importe quel champs ou alias dèfini dans `select_expression`. C'est èvaluè en dernier lieu, juste avant que les èlèments ne soient envoyès au client, sans aucune optimisation. N'utilisez pas `HAVING` pour des èlèments qui devraient être dans la clause `WHERE`. Par exemple, n'èchrivez pas ceci :

```
mysql> SELECT nom_de_colonne FROM nom_de_table HAVING nom_de_colonne > 0;
```

  Ecrivez plutôt cela :

```
mysql> SELECT nom_de_colonne FROM nom_de_table WHERE nom_de_colonne > 0;
```

  Dans les versions 3.22.5 et supèrieures de MySQL, vous pouvez aussi èchrire des requêtes ainsi :

```
mysql> SELECT user,MAX(salary) FROM users
    ->        GROUP BY user HAVING MAX(salary)>10;
```

  Dans des versions plus anciennes de MySQL, vous pouvez èchrire á la place :

```
mysql> SELECT user,MAX(salary) AS sum FROM users
    ->        group by user HAVING sum>10;
```

- Les options `DISTINCT`, `DISTINCTROW` et `ALL` indiquent quels enregistrements avec doublons doivent être retournès. Par dèfaut, c'est (`ALL`), retournant ainsi tous les enregistrements. `DISTINCT` et `DISTINCTROW` sont synonymes et indique que les doublons doivent être èliminès du rèsultat.
- Toutes les options commençant par `SQL_`, `STRAIGHT_JOIN`, et `HIGH_PRIORITY` sont des extensions MySQL de ANSI SQL.
- `HIGH_PRIORITY` donne á une commande `SELECT` une plus grande prioritè qu'une commande qui modifie une table. Vous devez l'utiliser seulement pour les requêtes qui sont trés rapides et qui doivent être effectuèes en premier lieu. Une requête `SELECT HIGH_PRIORITY` s'exècutera sur une table verrouillèe en lecture même si une commande de mise á jour attend que la table soit libèrèe.
- `SQL_BIG_RESULT` peut être utilisè avec `GROUP BY` ou `DISTINCT` pour indiquer á l'optimiseur que le rèsultat comportera beaucoup d'enregistrements. Dans ce cas, MySQL utilisera si besoin directement les bases temporaires stockèes sur le disque. MySQL prèfèrera, dans ce cas, trier que d'obtenir une table temporaire avec une clè sur les èlèments du `GROUP BY`.

- SQL_BUFFER_RESULT forcera le rèsultat á être stockè dans une table temporaire. Ceci va aider MySQL á libèrer plus tôt les verrous des tables et aidera aussi dans les cas ou l'envoi du rèsultat au client prend un temps assez consèquent.
- SQL_SMALL_RESULT, une option spècifique á MySQL, peut être utilisèe avec GROUP BY ou DISTINCT pour indiquer á l'optimiseur que le rèsultat sera petit. Dans ce cas, MySQL utilise des tables temporaires rapides pour stocker la table rèsultante plutôt que d'utiliser le tri. Dans MySQL 3.23, ceci n'est normalement pas nècessaire.
- SQL_CALC_FOUND_ROWS (version 4.0.0 et supèrieure) indique á MySQL de calculer combien d'enregistrements seront dans le jeu de rèsultats, indèpendamment de n'importe quelle clause LIMIT. Le nombre d'enregistrements peut alors être obtenu avec SELECT FOUND_ROWS(). Voir Section 6.3.6.2 [Miscellaneous functions], page 472.

  Veuillez noter que dans les versions antèrieures á 4.1.0, ceci ne fonctionne pas avec LIMIT 0, qui est optimisè pour retourner le rèsultat instantanèment (retournant bien entendu un nombre d'enregistrements ègal á 0). Voir Section 5.2.8 [LIMIT optimisation], page 374.
- SQL_CACHE indique á MySQL de stocker le rèsultat de la requête dans le cache de requêtes si vous utilisez QUERY_CACHE_TYPE=2 (DEMAND). Voir Section 6.9 [Query Cache], page 527.
- SQL_NO_CACHE indique á MySQL de ne pas permettre au rèsultat de la requête d'être stockè dans le cache de requêtes. Voir Section 6.9 [Query Cache], page 527.
- Si vous utilisez GROUP BY, les enregistrements en sortie seront triès en accord avec le GROUP BY comme si vous aviez eu un ORDER BY sur tous les champs du GROUP BY. MySQL a ètendu GROUP BY de façon á ce que vous puissiez spècifier ASC et DESC lors du GROUP BY:

  ```
  SELECT a,COUNT(b) FROM test_table GROUP BY a DESC
  ```
- MySQL a ètendu l'utilisation de GROUP BY pour permettre de sèlectionner des champs qui ne sont pas mentionnès dans la clause GROUP BY. Si vous n'obtenez pas les rèsultats que vous attendez de votre requête, veuillez consulter la description de GROUP BY. Voir Section 6.3.7 [Group by functions], page 479.
- STRAIGHT_JOIN force l'optimiseur á joindre les tables dans l'ordre oú elles sont listèes dans la clause FROM. Vous pouvez utiliser cela pour accèlèrer une requête si l'optimiseur joint les tables dans un ordre qui n'est pas optimal. Voir Section 5.2.1 [EXPLAIN], page 363.
- La clause LIMIT peut être utilisèe pour limiter le nombre d'enregistrements retournès par la commande SELECT. LIMIT accepte un ou deux arguments numèriques. Ces arguments doivent être des entiers constants.

  Si deux arguments sont donnès, le premier indique le dècalage du premier enregistrement á retourner, le second donne le nombre maximum d'enregistrement á retourner. Le dècallage du premier enregistrement est 0 (pas 1):

  ```
  mysql> SELECT * FROM table LIMIT 5,10;  # Retourne les enregistrements 6 á 15
  ```

  Pour obtenir tous les enregistrement d'un certain dècallage jusqu'á la fin du rèsultat, vous pouvez utiliser -1 en tant que second paramétre:

  ```
  mysql> SELECT * FROM table LIMIT 95,-1; # Retourne les enregistrements de 96 jus
  ```

Si un seul argument est donnè, il indique le nombre maximum d'enregistrements á retourner :

```
mysql> SELECT * FROM table LIMIT 5;     # Retourne les 5 premiers enregistrement
```

Autrement dit, `LIMIT n` est èquivalent á `LIMIT 0,n`.

- La forme `SELECT ... INTO OUTFILE 'nom_fichier'` de `SELECT` ècrit les lignes sèlectionnèes dans un fichier. Le fichier est crèe sur le serveur et ne peut y être dèjá prèsent (cela permet entre autre d'èviter la destruction des tables et de fichiers tel que '`/etc/passwd`'). Vous devez avoir le droit `FILE` sur le serveur pour utiliser cette forme de `SELECT`.

  `SELECT ... INTO OUTFILE` á pour but principal de vous permettre de rèaliser des dumps rapides des tables sur la machine serveur. Si vous voulez crèer le fichier sur une autre machine, vous ne pouvez utiliser `SELECT ... INTO OUTFILE`. Dans ce cas lá, vous pouvez utiliser á la place un programme client comme `mysqldump --tab` ou `mysql -e "SELECT ..." > fichier` pour gènèrer le fichier.

  `SELECT ... INTO OUTFILE` est le complèment de `LOAD DATA INFILE`; La syntaxe pour la partie `export_options` de la requête se compose des mêmes clauses `FIELDS` et `LINES` que celles utilisèes avec la commande `LOAD DATA INFILE`. Voir Section 6.4.9 [LOAD DATA], page 497.

  Dans le fichier rèsultant, seul les caractéres suivants sont protègès par le caractére `ESCAPED BY` :

  - Le caractére `ESCAPED BY`
  - Les premier caractére de `FIELDS TERMINATED BY`
  - Les premier caractére de `LINES TERMINATED BY`

  De plus, `ASCII 0` est convertit en `ESCAPED BY` suivi de 0 (`ASCII 48`).

  La raison de ce qui prècéde est que vous devez **impèrativement** protèger chaque caractére `FIELDS TERMINATED BY`, `ESCAPED BY`, ou `LINES TERMINATED BY` pour assurer une relecture fiable du fichier. Le caractére `ASCII 0` est èchappè pour assurer la lisibilitè sur certains clients. Comme le fichier rèsultant ne se doit pas d'être syntaxiquement conforme á SQL, vous n'avez besoin d'èchapper rien d'autre.

  Voila un exemple de relecture de fichier au format utilisè par plusieurs anciens programmes.

  ```
  SELECT a,b,a+b INTO OUTFILE "/tmp/result.text"
  FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
  LINES TERMINATED BY "\n"
  FROM test_table;
  ```

- Si vous utilisez `INTO DUMPFILE` au lieu de `INTO OUTFILE`, MySQL n'ècrira qu'une seule ligne dans le fichier, sans aucun caractére de fin de ligne ou de colonne, ni d'èchappement. Cela est utile lorsque vous voulez enregistrer un blob dans un fichier.
- Notez que chaque fichier crèe par `INTO OUTFILE` et `INTO DUMPFILE` sera lisible par tout les utilisateurs ! La raison est que le serveur MySQL ne peut crèer de fichier appartenant á autre que l'utilisateur qui l'a mit en route. (vous devez èviter d'exècuter `mysqld` en tant que root), le fichier doit se composer de mot lisible pour que les donnèes puissent être rècupèrèes.

- Si vous utilisez la clause `FOR UPDATE` avec un gestionnaire de tables qui gére les verrous de lignes ou de pages, les lignes seront verrouillèes.

6.4.1.1 Syntaxe de `JOIN`

MySQL supporte les syntaxes suivantes de `JOIN` pour une utilisation dans les `SELECT` :

```
reference_table, reference_table
reference_table [CROSS] JOIN reference_table
reference_table INNER JOIN reference_table condition_jointure
reference_table STRAIGHT_JOIN reference_table
reference_table LEFT [OUTER] JOIN reference_table condition_jointure
reference_table LEFT [OUTER] JOIN reference_table
reference_table NATURAL [LEFT [OUTER]] JOIN reference_table
{ OJ reference_table LEFT OUTER JOIN reference_table ON expr_conditionnelle }
reference_table RIGHT [OUTER] JOIN reference_table condition_jointure
reference_table RIGHT [OUTER] JOIN reference_table
reference_table NATURAL [RIGHT [OUTER]] JOIN reference_table
```

oú `reference_table` est dèfinie de la maniére suivante :

```
nom_de_table [[AS] alias] [USE INDEX (liste_de_clefs)] [IGNORE INDEX (liste_de_clef
```

et `condition_jointure` est dèfinie comme suit :

```
ON expr_conditionnelle |
USING (column_list)
```

Gènèralement, vous ne devez avoir aucune condition, dans la partie `ON`, qui soit utilisèe pour spècifier les lignes que vous voulez obtenir en rèsultat. (il y'a des exceptions á cette régle). Si vous voulez restreindre les lignes rèsultantes, vous devez le faire dans la clause `WHERE`.

Notez que dans les versions antèrieures á la 3.23.17, `INNER JOIN` ne prenait pas en compte `condition_jointure` !

La derniére syntaxe de `LEFT OUTER JOIN` vue plus haut, n'existe que pour assurer la compatibilitè avec ODBC :

- On peut crèer un alias sur une rèfèrence de table en utilisant `nom_de_table AS alias_name` ou `nom_de_table alias_name` :

  ```
  mysql> SELECT t1.name, t2.salary FROM employee AS t1, info AS t2
      ->        WHERE t1.name = t2.name;
  ```

- La condition `ON` est de la même forme qu'une condition pouvant être utilisèe dans la clause `WHERE`.
- Si aucune ligne ne correspond dans la table de droite dans la partie `ON` ou `USING` du `LEFT JOIN`, une ligne avec toutes les colonnes mises á `NULL` est utilisè en remplacement. Vous pouvez utiliser ce fait pour trouver les enregistrements dans une table qui n'ont pas de correspondances dans une autre :

  ```
  mysql> SELECT table1.* FROM table1
      ->        LEFT JOIN table2 ON table1.id=table2.id
      ->        WHERE table2.id IS NULL;
  ```

  Cet exemple retourne toutes les lignes trouvèes dans `table1` avec une valeur de `id` qui n'est pas prèsente dans `table2` (autrement dit, toutes les lignes de `table1` sans

correspondances dans la table `table2`). Cela demande que `table2.id` soit dèclarè `NOT NULL`, bien sur. Voir Section 5.2.6 [LEFT JOIN optimisation], page 372.

- La clause `USING (column_list)` recense la liste des colonnes qui doivent exister dans les deux tables. Une clause `USING` comme :

  ```
  A LEFT JOIN B USING (C1,C2,C3,...)
  ```

  est dèfinie pour être sèmantiquement identique á une expression `ON` comme :

  ```
  A.C1=B.C1 AND A.C2=B.C2 AND A.C3=B.C3,...
  ```

- La jointure de deux tables avec `NATURAL [LEFT] JOIN` est dèfini pour être sèmantiquement èquivalent á un `INNER JOIN` ou un `LEFT JOIN` avec une clause `USING` qui nomme toutes les colonnes qui existent dans les deux tables.
- `INNER JOIN` et `,` (virgule) sont sèmentiquement èquivalents. Les deux opérent une jointure totale sur les tables utilisèes. Normalement, vous spècifiez les conditions de jointure dans la clause WHERE.
- `RIGHT JOIN` fonctionne de façon analogue á `LEFT JOIN`. Pour garder un code facilement portable, il est recommandè d'utiliser les `LEFT JOIN` á la place des `RIGHT JOIN`.
- `STRAIGHT_JOIN` est identique á `JOIN`, sauf que la table de gauche est toujours lues avant celle de droite. Cela peut être utilisè dans les cas (rares) oú l'optimiseur des jointures place les tables dans le mauvais ordre.
- A partir de la version 3.23.12 de MySQL, vous pouvez donner des indications á propos de l'index á utiliser lors de la lecture d'informations d'une table. C'est utile si `EXPLAIN` montre que MySQL utilise un mauvais index de la liste de ceux disponibles. En spècifiant `USE INDEX (liste_de_clefs)`, vous pouvez forcer MySQL á utiliser un index spècifique pour trouver les enregistrements dans la table. Une alternative rèside dans l'utilisation de `IGNORE INDEX (liste_de_clefs)` pour dire á MySQL de ne pas utiliser certains index.

  `USE/IGNORE KEY` sont des synonymes de `USE/IGNORE INDEX`.

Quelques exemples :

```
mysql> SELECT * FROM table1,table2 WHERE table1.id=table2.id;
mysql> SELECT * FROM table1 LEFT JOIN table2 ON table1.id=table2.id;
mysql> SELECT * FROM table1 LEFT JOIN table2 USING (id);
mysql> SELECT * FROM table1 LEFT JOIN table2 ON table1.id=table2.id
    ->          LEFT JOIN table3 ON table2.id=table3.id;
mysql> SELECT * FROM table1 USE INDEX (key1,key2)
    ->          WHERE key1=1 AND key2=2 AND key3=3;
mysql> SELECT * FROM table1 IGNORE INDEX (key3)
    ->          WHERE key1=1 AND key2=2 AND key3=3;
```

Voir Section 5.2.6 [`LEFT JOIN` optimisation], page 372.

6.4.1.2 Syntaxe de `UNION`

```
SELECT ...
UNION [ALL]
SELECT ...
  [UNION
```

```
    SELECT ...]
```

`UNION` est implèmentèe en MySQL 4.0.0.

`UNION` est utilisè pour combiner le rèsultat de plusieurs requêtes `SELECT` en un seul rèsultat.

Les colonnes listèes dans la partie select_expression du `SELECT` doivent être du même type. Les noms de colonnes utilisès dans le premier `SELECT` seront utilisè comme nom de champs pour les rèsultats retournès.

Les commandes `SELECT` sont des select normaux, mais avec les restrictions suivantes :

- Seule la derniére commande `SELECT` peut avoir une clause `INTO OUTFILE`.

Si vous n'utilisez pas le mot clef `ALL` pour l'`UNION`, toutes les lignes retournèes seront uniques, comme si vous aviez fait un `DISTINCT` pour l'ensemble du rèsultat. Si vous spècifiez `ALL`, vous aurez alors tout les rèsultats retournès par toutes les commandes `SELECT`.

Si vous voulez utiliser un `ORDER BY` pour le rèsultat final de `UNION`, vous devez utiliser des parenthéses :

```
(SELECT a FROM nom_de_table WHERE a=10 AND B=1 ORDER BY a LIMIT 10)
UNION
(SELECT a FROM nom_de_table WHERE a=11 AND B=2 ORDER BY a LIMIT 10)
ORDER BY a;
```

6.4.2 Syntaxe de HANDLER

```
HANDLER nom_de_table OPEN [ AS alias ]
HANDLER nom_de_table READ nom_index { = | >= | <= | < } (value1,value2,...)
    [ WHERE ... ] [LIMIT ... ]
HANDLER nom_de_table READ nom_index { FIRST | NEXT | PREV | LAST }
    [ WHERE ... ] [LIMIT ... ]
HANDLER nom_de_table READ { FIRST | NEXT }
    [ WHERE ... ] [LIMIT ... ]
HANDLER nom_de_table CLOSE
```

La commande `HANDLER` fournit un accés direct á l'interface de gestion de la table `MyISAM`.

La premiére forme de `HANDLER` ouvre la table, la rendant accessible via la requête `HANDLER ... READ` qui la suit. Cette objet table n'est pas partagè par les autres threads et ne sera refermè que si le thread appelle `HANDLER nom_de_table CLOSE` ou que celui ci se termine.

La seconde forme rècupére une ligne (ou plus, á spècifier dans la clause `LIMIT`) oú l'index spècifiè remplit les conditions et oú la clause `WHERE` est rèpondue. Si l'index se compose de plusieurs parties, (s'ètend sur plusieurs colonnes) les valeurs sont spècifiées dans une liste sèparèe par des virgules, fournir des valeurs pour quelques premiéres colonnes est possible.

La troisiéme forme rècupére une ligne (ou plus, á spècifier dans la clause `LIMIT`) de la table dans l'ordre de l'index, qui rèpond á la clause `WHERE`.

La quatriéme forme (sans spècifications relatives á l'index) rècupére une ligne (ou plus, á spècifier dans la clause `LIMIT`) de la table dans un ordre naturel des lignes (comme stockè dans le fichier de donnèes) qui correspond á la condition `WHERE`. C'est plus rapide que `HANDLER nom_de_table READ nom_index` quand une lecture entiére de la table est requise.

`HANDLER ... CLOSE` ferme une table qui a ètè ouverte avec `HANDLER ... OPEN`.

HANDLER est en quelque sorte une commande bas-niveau. Par exemple, elle ne propose pas de consistance. En clair, HANDLER ... OPEN ne se base **PAS** sur une image de la table, et ne verrouille **PAS** la table. Cela signifie qu'aprés l'exècution d'une requête HANDLER ... OPEN, les donnèes de la table peuvent être modifièes (par ce ou un autre thread) et ces modifications peuvent apparaître partiellement dans les lectures de HANDLER ... NEXT ou HANDLER ... PREV.

Les raisons d'utiliser cette interface plutôt que les commandes MySQL usuelles sont :

- Plus rapide qu'un SELECT car :
  - Un pointeur sur table dèdiè est allouè au thread dans HANDLER open.
  - Il y'a moins de traitements.
  - Pas de pertes de temps en optimisation ou vèrifications de requêtes.
  - La table utilisèe n'a pas besoin d'être verrouillèe entre deux requêtes de gestion.
  - L'interface de gestion n'a pas á fournir une vue consistante des donnèes (par exemple, les lectures corrompues sont autorisèes), ce qui permet au gestionnaire d'effectuer des optimisations que SQL ne permet pas.
- Cela facilite le port des applications qui utilisent l'interface ISAM pour MySQL.
- Cela permet de traverse plus facilement la base de donnèes qu'avec SQL (dans certains cas, cette opèration est impossible avec SQL). L'interface de gestion amène une façon plus naturelle de manipuler les donnèes lorsque vous travaillez avec des applications qui proposent une interface interactive entre l'utilisateur et la base de donnèes.

6.4.3 Syntaxe de INSERT

```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
[INTO] nom_de_table [(nom_colonne,...)]
VALUES ((expression | DEFAULT),...),(...),...
ou  INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
[INTO] nom_de_table [(nom_colonne,...)]
SELECT ...
ou  INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
[INTO] nom_de_table
:    SET nom_colonne=(expression | DEFAULT), ...
```

INSERT insére une nouvelle ligne dans une table existante. La syntaxe INSERT ... VALUES insére une ligne á partir de valeurs explicitement fournies. La syntaxe INSERT ... SELECT insére des valeurs á partir d'une autre table. La syntaxe INSERT ... VALUES avec plusieurs valeurs est supportèes á partir de MySQL Version 3.22.5 ou supèrieure. la syntaxe nom_colonne=expression est supportèe á partir de la version 3.22.10 de MySQL.

nom_de_table est le nom de la table dans laquelle les valeurs seront insèrèes. La liste de noms de colonne ou la clause SET indiquent les colonnes qui seront affectèes:

- Si vous ne spècifiez pas de liste de colonnes avec INSERT ... VALUES ou INSERT ... SELECT, les valeurs pour toutes les colonnes doivent être fournies dans la clause VALUES() ou par la commande SELECT. Si vous ne connaissez pas l'ordre des colonnes, utilisez la commande DESCRIBE nom_de_table pour le connaître.

- A chaque fois qu'on ne donne pas explicitement une valeur pour une colonne, celle prend la valeur par dèfaut. Par exemple, si on dèfinit une liste de colonnes qui ne compte pas toutes les colonnes de la tables, toutes les colonnes qui ne sont pas nommèes prendront leur valeur par dèfaut. La dèfinition de la valeur par dèfaut se fait avec Section 6.5.3 [`CREATE TABLE`], page 504.

  Il est ègalement possible d'utiliser le mot `DEFAULT` pour qu'une colonne prenne sa valeur par dèfaut. (Nouveautè de MySQL 4.0.3) Il est ainsi plus simple d'ècrire des requêtes `INSERT` qui donnent des valeurs á toutes les colonnes sauf quelques unes, car cela permet de ne pas avoir á ècrire la liste incompléte de `VALUES()` (une liste qui ne comprend pas une valeur pour chaque colonne de la table). Sans cela, il faudrait ècrire la liste des noms de colonnes correspondant a chaque valeur de la liste `VALUES()`.

  MySQL a toujours une valeur par dèfaut pour chaque champs. C'est obligatoire pour MySQL pour pouvoir fonctionner aussi bien avec des tables supportant les transactions qu'avec des tables ne les supportant pas.

  Nous pensons que le contrôle du contenu des champs devrait être fait pas l'application et non par le serveur de base de donnèes.

- Une `expression` peut faire rèfèrence á n'importe quelle colonne qui a ètè dèfinie prècèdemment dans une liste de valeurs. Par exemple, on peut dire ceci:

  ```
  mysql> INSERT INTO nom_de_table (col1,col2) VALUES(15,col1*2);
  ```

  Mais pas cela:

  ```
  mysql> INSERT INTO nom_de_table (col1,col2) VALUES(col2*2,15);
  ```

- Si on spècifie le mot `LOW_PRIORITY`, l'exècution de `INSERT` sera retardè jusqu'á ce qu'il n'y ait plus de clients qui lisent la table. Dans ce cas le client doit attendre jusqu'á la fin de l'opèration d'insertion, ce qui peut prendre beaucoup de temps si la table est frèquemment accèdèe. C'est la grande diffèrence avec `INSERT DELAYED`, qui laisse le client continuer tout de suite. Voir Section 6.4.4 [INSERT DELAYED], page 491. On peut remarquer que `LOW_PRIORITY` ne devrait en principe pas être utiliser avec des tables de type `MyISAM`, ètant donnè que celle si n'autorisent pas les inserts concurrents. Voir Section 7.1 [MyISAM], page 532.
- Si on spècifie le mot `IGNORE` dans un `INSERT` avec les valeurs de plusieurs lignes, chaque ligne qui qui ferait doublon avec une clè `PRIMARY` ou `UNIQUE` existante dans la table sera ignorè et ne sera pas insèrèe. Si on ne spècifie pas `IGNORE`, l'insert est abandonnè si quelque ligne que ce soit fait doublon avec une clè existante. La fonction `mysql_info()` de l'API C permet de savoir combien de lignes ont ètè insèrèes dans la table.
- Si MySQL a ètè configurè avec l'option `DONT_USE_DEFAULT_FIELDS`, les opèrations `INSERT` gènére une erreur tant que les valeurs de toutes les colonnes qui attendent une valeur autre que `NULL` ne sont pas spècifièes explicitement. Voir Section 2.3.3 [les options de `configure`], page 88.
- Il est possible de rècupèrer la valeur utilisèe pour une colonne `AUTO_INCREMENT` avec la fonction `mysql_insert_id`. Voir Section 8.4.3.30 [`mysql_insert_id()`], page 635.

Si on fait une opèration `INSERT ... SELECT` ou `INSERT ... VALUES` avec plusieurs listes de valeurs, la fonction `mysql_info()` de l'API C permet d'obtenir des informations relatives á la requête. Le format de la chaîne d'information est la suivante:

```
Records: 100 Duplicates: 0 Warnings: 0
```

`Duplicates` indique le nombre de lignes qui n'ont pas peut être insèrèes pour cause de conflit avec une clè unique existante. `Warnings` indique le nombre de tentatives d'inserts de valeurs dans une colonne qui ont gènèrè des problémes. Les `Warnings` peuvent apparaître dans les conditions suivantes:

- Insertion de `NULL` dans une colonne dèclarèe `NOT NULL`. Cette colonne a ètè enregistrèe avec sa valeur par dèfaut.
- Enregistrement dans une colonne numèrique d'une valeur qui dèborde de la taille de la colonnes. Cette valeur a ètè tronquèe á l'extrèmitè la plus adaptèe de la colonne.
- Attribution á une colonne numèrique d'une valeur telle que `'10.34 a'`. Celle valeur refusèe est sèparèe, et la partie numèrique rèsultante est insèrèe. Si cette valeur n'a pas une valeur numèrique sensèe, la valeur `0` est insèrèe.
- L'insertion d'une chaîne dans une colonne `CHAR`, `VARCHAR`, `TEXT`, ou `BLOB` qui dèpasse la taille maximale de la colonne. La valeur est tronquèe á la taille maximale de la colonne.
- L'insertion d'une valeur illègale pour une colonne de type date ou time. La colonne est alors enregistrèe avec la valeur de zero approprièe pour le type.

6.4.3.1 Syntaxe de `INSERT ... SELECT`

```
INSERT [LOW_PRIORITY] [IGNORE] [INTO] nom_de_la_table [(liste des colonnes)] SELECT
```

La requête `INSERT ... SELECT` permet de rapidement insèrer dans une table un grand nombre de lignes d'une ou plusieurs autres tables.

```
INSERT INTO tblTemp2 (fldID) SELECT tblTemp1.fldOrder_ID FROM tblTemp1 WHERE
tblTemp1.fldOrder_ID > 100;
```

Les conditions suivantes s'appliquent á la requête `INSERT ... SELECT`:

- La table de destination de la requête `INSERT` ne peut apparaître dans la clause `FROM` de la partie `SELECT` de la requête car il est interdit par le ANSI SQL de lire la table dans laquelle on est en train de faire un insert. (le probléme est que le `SELECT` pourrait trouver des enregistrements qui aurait ètè insèrès auparavant dans la même exècution. L'utilisation de "subselect" peut rendre la situation confuse !)
- Les colonnes `AUTO_INCREMENT` fonctionnent comme d'habitude.
- Il est possible d'utiliser la fonction `mysql_info()` de l'API C pour obtenir des informations sur la requête. Voir Section 6.4.3 [INSERT], page 489.
- Pour s'assurer que les journaux des modifications ou les journaux binaires puissent être utilisès pour re-crèer les tables originales, MySQL n'autorise pas les inserts concurrents pendant `INSERT ... SELECT`.

Il est bien sûr possible d'utiliser `REPLACE` á la place de `INSERT` pour remplacer les anciennes lignes.

6.4.4 Syntaxe de `INSERT DELAYED`

```
INSERT DELAYED ...
```

L'option `DELAYED` de la commande `INSERT` est une option spècifique á MySQL trés utile si vos clients ne peuvent pas attendre que `INSERT` se termine. C'est un probléme frèquent

quand on utilise MySQL pour des logs, mais aussi quand on utilise souvent des commandes `SELECT` ou `UPDATE` qui prennent beaucoup de temps. `DELAYED` a ètè ajoutè á MySQL dans la version 3.22.15. C'est une extension de MySQL au ANSI SQL 92.

`INSERT DELAYED` fonctionne avec les tables `ISAM` et `MyISAM`. On peut noter que, les tables `MyISAM` supportant les `SELECT` et les `INSERT` concurrents, s'il n'y a pas de blocs libres au milieu du fichier de donnèes il est vraiment conseillè d'utiliser `INSERT DELAYED` avec les tables `MyISAM`. Voir Section 7.1 [MyISAM], page 532.

En utilisant `INSERT DELAYED`, le client reçoit immèdiatement un aquitement, et la ligne sera insèrèe quand la table ne sera plus utilisèe par un autre thread.

Un autre avantage de `INSERT DELAYED` est que les insertions des clients sont regroupès, et ècrits d'un seul bloc. C'est beaucoup plus rapide que de faire des insertions sèparès.

Actuellement, les lignes en attente sont uniquement stockèes en mèmoire tant qu'elle ne sont pas insèrèes dans la table. Cela signifie que si on tue `mysqld` violemment, (`kill -9`) ou si `mysqld` meurt accidentellement, toutes les lignes en attente qui n'auront pas ètè ècrites sur le disque seront perdues !

Les paragraphes suivants dècrivent en dètail ce qu'il se passe quand on utilise l'option `DELAYED` dans une requête `INSERT` ou `REPLACE`. Dans cette description, "thread" est un thread qui reçoit une commande `INSERT DELAYED` ans "handler" est un thread qui gére toutes les opèrations de `INSERT DELAYED` pour une table donnèe.

- Quand un thread exècute une opèration `DELAYED` sur une table, un thread de gestion est crèè pour exècuter toutes les opèrations `DELAYED` pour cette table - si ce thread de gestion n'existe pas.
- Le thread vèrifie que a dèjá reçu un verrou `DELAYED`; sinon, il dit au thread de gestion de le faire. le verrou `DELAYED` peut être obtenu même si d'autres threads ont des verrous `READ` ou `WRITE` sur la table. Cependant le gestionnaire attendra que tous les verrous `ALTER TABLE` ou `FLUSH TABLES` soient finis pour s'assurer que la structure de la table est á jour.
- Le thread exècute une opèration `INSERT`, mais plutôt que d'ècrire la ligne dans la table, il va placer une copie de la ligne finale dans une file d'attente gèrèe par le thread de gestion. Le programme client est avertit de toutes les erreurs de syntaxe.
- Le client ne peut pas faire de rapport sur le nombre de duplicata ou sur la valeur de `AUTO_INCREMENT` de la ligne enregistrèe; il ne peut pas les obtenir du serveur, car le `INSERT` est validè avant que l'opèration d'insert n'ait ètè effectuèe. Si vous utilisez l' API C, la fonction `mysql_info()` ne retourne pas de valeur intèressante, pour la même raison.
- Le journal de modification est mis á jour par le thread de gestion au moment oú la ligne est insèrèe dans la table. Si plusieurs lignes sont insèrèes en même temps, le journal des modifications est mis á jour quand la premiére ligne est insèrèe.
- Une fois que toutes les lignes `delayed_insert_limit` sont ècrites, le gestionnaire vèrifie si des requêtes `SELECT` sont en attente, et si c'est le cas, il leur permet de s'exècuter avant de continuer.
- Quand le thread de gestion n'a plus de ligne dans sa file, la table est dèverrouillèe. Si aucun `INSERT DELAYED` n'est reçu avant `delayed_insert_timeout` secondes, le gestionnaire s'arrête.

- Si plus de `delayed_queue_size` lignes sont dèjá en attente d'un gestionnaire de file donnè, le thread qui demande le `INSERT DELAYED` doit attendre qu'il y ait une place dans la file. Cela permet d'être sûr que `mysqld` n'utilisera pas toute la mèmoire pour la mèmoire des files d'attente d'insertions retardès.
- Le thread de gestion apparaîtra dans la liste des processus de MySQL avec `delayed_insert` dans la colonne `Command`. Il sera tuè si on exècute une commande `FLUSH TABLES` ou si on le tue avec `KILL thread_id`. Cependant, il commencera par stocker toutes les lignes en attente dans la table avant de sortir. Pendant ce temps, il n'acceptera aucune commande `INSERT` d'aucun autre thread. Si on exècute une commande `INSERT DELAYED` aprés cela, un nouveau thread de gestion sera crèè.

  Il faut noter que les commandes `INSERT DELAYED` ont une plus grande prioritè que les commandes `INSERT` normales si un gestionnaire de `INSERT DELAYED` existe dèjá! les autres commandes de modification devront attendre que la file d'attente de `INSERT DELAYED` soit vide, que quelqu'un tue le gestionnaire (avec `KILL thread_id`), ou que quelqu'un exècute `FLUSH TABLES`..
- Les variables suivantes fournissent des informations relatives á la commande `INSERT DELAYED` :

Variable	Signification
`Delayed_insert_threads`	Nombre de threads de gestion
`Delayed_writes`	Nombre de lignes ècrites avec `INSERT DELAYED`
`Not_flushed_delayed_rows`	Nombre de lignes en attente d'être ècrites.

  On peut voir ces variables avec la commande `SHOW STATUS` ou en exècutant la commande `mysqladmin extended-status`.

Il faut noter que `INSERT DELAYED` est plus lent qu'un INSERT normal si la table n'est pas utilisèe. L'utilisation d'un thread de gestion sèparè pour chaque table sur lesquelles on utilise `INSERT DELAYED` rajoute ègalement une surcharge au serveur. Ce qui signifie qu'il vaut mieux utiliser `INSERT DELAYED` uniquement quand c'est vraiment nècessaire!

6.4.5 Syntaxe de `UPDATE`

```
UPDATE [LOW_PRIORITY] [IGNORE] nom_de_table
    SET nom_colonne1=expr1 [, nom_colonne2=expr2, ...]
    [WHERE where_definition]
    [ORDER BY ...]
    [LIMIT #]
```

`UPDATE` met á jour des enregistrements dans une tables avec de nouvelles valeurs. La clause `SET` indique les colonnes á modifier et les valeurs á leur donner. La clause `WHERE`, si fournie, spècifie les enregistrements á mettre á jour. Sinon, tous les enregistrements sont mis á jour. Si la clause `ORDER BY` est fournie, les enregistrements seront mis á jour dans l'ordre spècifiè.

Si vous spècifiez le mot clef `LOW_PRIORITY`, l'exècution de l'`UPDATE` sera repoussè jusqu'á ce que aucun client ne lise plus de la table.

Si vous spècifiez le mot clef `IGNORE`, la mise á jour ne s'interrompra pas même si on rencontre des problémes d'unicitè de clefs durant l'opèration. Les enregistrements posant problémes ne seront pas mis á jour.

Si vous accèdez á une colonne d'une table dans une expression, `UPDATE` utilisera la valeur courante de la colonne. Par exemple, la requête suivante ajoute une année á l'âge actuel de tout le monde :

```
mysql> UPDATE persondata SET age=age+1;
```

Les requêtes `UPDATE` sont èvaluèes de gauche á droite. Par exemple, la requête suivante double la valeur de la colonnes âge, puis l'incrèmente :

```
mysql> UPDATE persondata SET age=age*2, age=age+1;
```

Si vous changez la valeur d'une colonne en lui spècifiant sa valeur actuelle, MySQL s'en aperçoit et ne fait pas la mise á jour.

`UPDATE` retourne le nombre d'enregistrements ayant changè. Depuis la version 3.22 de MySQL, la fonction `mysql_info()` de l'API C retourne le nombre de colonnes qui correspondaient, le nombre de colonnes mises á jour et le nombre d'erreurs gènèrèes pendant l'`UPDATE`.

Dans la version 3.23 de MySQL, vous pouvez utilisez le code `LIMIT #` pour vous assurer que seul un nombre d'enregistrements bien prècis est changè.

A partir de la version 4.0.4 de MySQL, vous pouvez aussi effectuer un `UPDATE` qui se base sur plusieurs tables :

```
UPDATE items,month SET items.price=month.price
WHERE items.id=month.id;
```

6.4.6 Syntaxe de `DELETE`

```
DELETE [LOW_PRIORITY] [QUICK] FROM nom_de_table
       [WHERE clause_where]
       [ORDER BY ...]
       [LIMIT lignes]

ou

DELETE [LOW_PRIORITY] [QUICK] nom_de_table[.*] [,nom_de_table[.*] ...]
       FROM table-references
       [WHERE clause_where]

ou

DELETE [LOW_PRIORITY] [QUICK]
       FROM nom_de_table[.*], [nom_de_table[.*] ...]
       USING table-references
       [WHERE clause_where]
```

`DELETE` efface les enregistrements de `nom_de_table` qui satisfont la condition donnèe par `clause_where`, et retourne le nombre d'enregistrements effacès.

Si vous exècutez un `DELETE` sans clause `WHERE`, tous les enregistrements sont effacès. Si vous le faites en mode `AUTOCOMMIT` cela aura le même effet qu'un `TRUNCATE`. Voir Section 6.4.7 [`TRUNCATE`], page 496. Avec MySQL 3.23, `DELETE` sans clause `WHERE` retournera zèro comme nombre d'enregistrements affectès.

Si vous voulez vraiment savoir combien d'enregistrements ont ètè effacès quand vous videz une table, et que vous êtes prêts á souffrir d'un lèger ralentissement, vous pouvez utiliser une requête `DELETE` de ce genre :

```
mysql> DELETE FROM nom_de_table WHERE 1>0;
```

Notez que c'est plus lent que `DELETE FROM nom_de_table` sans clause `WHERE`, parce que cela efface un enregistrement á la fois.

Si vous spècifiez le mot clef `LOW_PRIORITY`, l'exècution du `DELETE` sera repoussèe jusqu'á ce qu'aucun client ne lise plus de la table.

Si vous spècifiez le mot `QUICK`, le handler de la table will not merge index leaves during delete, which may speed up certain kind of deletes.

Dans les tables de type `MyISAM`, les enregistrements effacès sont maintenus dans une liste liée et les requêtes `INSERT` suivantes rèutilisent les vieux emplacements. Pour recouvrir l'espace inutilisè ou rèduire la taille des fichiers, utilisez la commande `OPTIMIZE TABLE` ou l'utilitaire `myisamchk` pour rèorganiser les tables. `OPTIMIZE TABLE` est plus simple, mais `myisamchk` est plus rapide. Voyez Section 4.5.1 [`OPTIMIZE TABLE`], page 264 et Section 4.4.6.10 [Optimisation], page 257.

La premiére suppression multi-tables est supportèe á partir de MySQL 4.0.0. La premiére suppression multi-tables est supportèe á partir de MySQL 4.0.2.

L'idèe est que seul les lignes concordante dans les tables ènumèrèes **avant** le `FROM` ou avant la clause `USING` sont effacès. Le but est de pouvoir effacer des lignes de plusieurs tables en même temps tout en ayant d'autres tables pour les recherches.

Le code `.*` aprés les noms de tables n'est prèsent que pour assurer la compatibilitè avec `Access`:

```
DELETE t1,t2 FROM t1,t2,t3 WHERE t1.id=t2.id AND t2.id=t3.id

ou

DELETE FROM t1,t2 USING t1,t2,t3 WHERE t1.id=t2.id AND t2.id=t3.id
```

Dans les cas prècèdents, nous n'avons supprimè les lignes correspondantes que dans les tables `t1` et `t2`.

`ORDER BY` et l'utilisation de plusieurs tables dans une requête `DELETE` est supportè en MySQL 4.0.

Si une clause `ORDER BY` est utilisèe, les enregistrements seront effacès dans cet ordre. Ceci n'est vraiment intèressant qu'en conjonction avec `LIMIT`. Par exemple:

```
DELETE FROM unlog
WHERE user = 'jcole'
ORDER BY timestamp
LIMIT 1
```

Cela effacera la plus vielle entrèe (en se basant sur `timestamp`) oú les enregistrements satisfont la clause `WHERE`.

L'option `LIMIT lignes` spècifique á MySQL pour `DELETE` donne au serveur le nombre maximal de lignes á être effacè avant que le contrôle ne revienne au client. Cela peut être utilisè pour s'assurer qu'une commande `DELETE` spècifique ne prenne pas trop de temps. Vous

pouvez rèpèter la commande jusqu'á ce que le nombre de lignes affectèes soit infèrieur á la valeur passèe á l'option `LIMIT` .

6.4.7 Syntaxe de `TRUNCATE`

```
TRUNCATE TABLE nom_de_table
```

Dans la version 3.23, `TRUNCATE TABLE` est èquivalent á `COMMIT ; DELETE FROM nom_de_table`. Voir Section 6.4.6 [DELETE], page 494.

`TRUNCATE TABLE` diffère de `DELETE FROM ...` des façons suivantes :

- Implèmentèe comme une destruction/crèation de table, ce qui accèlére la suppression des enregistrements.
- Ne respecte pas les transactions. Vous aurez des erreurs si vous avez une transaction active ou une table protègèe en ècriture.
- Ne retourne pas le nombre de lignes effacèes.
- Tant que le fichier de dèfinition '`nom_de_table.frm`' est valide, la table peut être recrèèe, me si les donnèes ou un index a ètè corrompu.

`TRUNCATE` est une extension Oracle SQL.

6.4.8 Syntaxe de `REPLACE`

```
    REPLACE [LOW_PRIORITY | DELAYED]
        [INTO] nom_de_table [(nom_de_colonne,...)]
        VALUES (expression,...),(...),...
ou  REPLACE [LOW_PRIORITY | DELAYED]
        [INTO] nom_de_table [(nom_de_colonne,...)]
        SELECT ...
ou  REPLACE [LOW_PRIORITY | DELAYED]
        [INTO] nom_de_table
        SET col_name=expression, nom_de_colonne=expression,...
```

`REPLACE` fonctionne exactement comme `INSERT`, sauf que si une vieille ligne dans la table á la même valeur qu'une nouvelle pour un index `UNIQUE` ou une `PRIMARY KEY`, la vielle ligne sera effacèe avant que la nouvelle ne soit insèrèe. Voir Section 6.4.3 [INSERT], page 489.

En d'autres termes, vous ne pouvez pas accèder aux valeurs de l'ancienne ligne á partir d'une requête `REPLACE`. Dans quelques vieilles versions de MySQL, il apparaît que c'ètait possible, mais c'etait un dysfonctionnement qui a ètè corrigè depuis.

Pour utiliser `REPLACE` vous devez avoir les priviléges `INSERT` et `DELETE` sur la table.

Quand vous utilisez une commande `REPLACE`, `mysql_affected_rows()` retournera 2 si une nouvelle ligne en remplace une existante, et cela parce qu'il y'aura eu une insertion puis une suppression.

Cela aide á savoir si `REPLACE` a ajoutè ou a remplacè une ligne : Testez si le nombre de lignes affectèes est ègal á 1 (ajout) ou s'il est ègal á 2 (remplacement).

Notez que si vous n'utilisez pas un index `UNIQUE` ou une `PRIMARY KEY`, utiliser un `REPLACE` n'a pas de sens vu que cela revient á utiliser un `INSERT`.

6.4.9 Syntaxe de LOAD DATA INFILE

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'nom_de_fichier.txt'
    [REPLACE | IGNORE]
    INTO TABLE nom_de_table
    [FIELDS
        [TERMINATED BY '\t']
        [[OPTIONALLY] ENCLOSED BY '']
        [ESCAPED BY '\\' ]
    ]
    [LINES TERMINATED BY '\n']
    [IGNORE nombre LINES]
    [(nom_de_colonne,...)]
```

La commande `LOAD DATA INFILE` lit les lignes dans un fichier texte et les insére á trés grande vitesse. Si le mot clef `LOCAL` est spècifiè, le fichier sera lu sur la machine du client. Si le mot clef `LOCAL` n'est pas spècifiè, le fichier doit se trouver sur le serveur (`LOCAL` est disponible á partir de la version 3.22.6 de MySQL).

Pour des raisons de sècuritè, lorsque les fichiers sont lus sur le serveur, ils doivent se trouver dans le rèpertoire de la base de donnèes courante, ou bien être lisible par tous. Pour utiliser la commande `LOAD DATA INFILE` sur des fichiers du serveur, vous devez avoir le droit de `FILE` sur le serveur. Voir Section 4.2.7 [Privileges provided], page 212.

Dans les versions 3.23.49 et 4.0.2 de MySQL, `LOCAL` ne fonctionnera que si vous n'avez pas dèmarrè `mysqld` avec l'option `--local-infile=0` ou si vous n'avez pas activè le support de `LOCAL` pour votre client. Voir Section 4.2.4 [LOAD DATA LOCAL], page 209.

Si vous spècifiez le mot clef `LOW_PRIORITY`, l'exècution de la commande `LOAD DATA` est ajournèe jusqu'á ce qu'aucun client ne lise plus de la table.

Si vous spècifiez le mot clef `CONCURRENT` avec un table au format `MyISAM`, les autres threads pourront accèder á la table durant l'exècution de la commande `LOAD DATA`. L'utilisation de cette option ralentira un peu les performances de `LOAD DATA` même si aucun thread n'utilise la table en même si aucun autre thread n'accéde á la table en même temps.

L'utilisation de `LOCAL` sera un peu plus lente que laisser le serveur accèder directement au fichiers, car le contenu du fichier devra transiter du client jusqu'au serveur. D'un autre côtè, vous n'aurez pas besoin du droit `FILE` pour charger les fichiers locaux.

Si vous utilisez une version de MySQL antèrieure á la 3.23.24 vous ne pouvez lire á partir d'un FIFO avec `LOAD DATA INFILE`. Si vous avez besoin de lire á partir d'un FIFO (par exemple la sortie de gunzip), utilisez `LOAD DATA LOCAL INFILE`.

Vous pouvez aussi charger des fichiers de donnèes en utilisant l'utilitaire `mysqlimport`; Il opére en envoyant la commande `LOAD DATA INFILE` au serveur. L'option `--local` fais que `mysqlimport` lit les fichiers de donnèes chez le client. Vous pouvez spècifier l'option `--compress` pour avoir de meilleurs performances avec les connexions lentes si le client et le serveur supportent le protocole compressè.

Lorsque les fichiers de donnèes sont sur le serveur, celui çi utilise les régles suivantes :

- Si un chemin absolu est fourni, le serveur utilise le chemin tel quel.
- Si un chemin relatif est fourni, avec un ou plusieurs èlèments de dossiers, le serveur recherche le fichier relativement á son dossier de donnèes.

- Si le fichier n'a pas d'èlèments de dossier, le serveur recherche les donnèes dans le dossier de base de donnèes courante.

Notez que ces régles font qu'un fichier tel que '`./myfile.txt`' est lu dans le dossier de donnèes du serveur, alors que s'il est nommè '`myfile.txt`', il sera lu dans le dossier de base de donnèes courante. Par exemple, la commande `LOAD DATA` suivante lit le fichier '`donnees.txt`' dans le dossier de la base `db1` car `db1` est la base de donnèes courante, même si la commande charge explicitement le fichier dans la base de donnèes `db2` :

```
mysql> USE db1;
mysql> LOAD DATA INFILE "donnees.txt" INTO TABLE db2.ma_table;
```

Les mots rèservès `REPLACE` et `IGNORE` contrôlent la mèthode d'insertion de lignes lorsque des doublons apparaissent pour les clès uniques. Si vous spècifiez `REPLACE`, les nouvelles lignes remplaceront les anciennes. Si vous spècifiez `IGNORE`, les nouvelles lignes seront ignorèes. Si vous ne spècifiez pas cette option, une erreur sera gènèrèe á chaque doublon, et le reste du fichier sera ignorè.

Si vous chargez un fichier sur votre machine client avec l'option `LOCAL`, le serveur ne peut pas interrompre la transmission du fichier au milieu de l'opèration : par dèfaut, il utilisera l'option `IGNORE`.

Si vous utilisez `LOAD DATA INFILE` sur une table vide de type `MyISAM`, tous les index non-uniques seront crèès dans un processus sèparè (tout comme `REPAIR`). Cela rend `LOAD DATA INFILE` beaucoup plus rapide si vous avez plusieurs index.

`LOAD DATA INFILE` est le complèmentaire de `SELECT ... INTO OUTFILE`. Voir Section 6.4.1 [`SELECT`], page 481. Pour ècrire des donnèes depuis une table dans un fichier, utilisez `SELECT ... INTO OUTFILE`. Pour lire les donnèes dans la table, utilisez `LOAD DATA INFILE`. La syntaxe des clauses `FIELDS` et `LINES` est la même pour les deux commandes. Ces deux clauses sont optionelles, mais `FIELDS` doit prècèder `LINES`, si les deux sont specifièes.

Si vous spècifiez la clause `FIELDS`, les sous-clauses `TERMINATED BY`, `[OPTIONALLY] ENCLOSED BY`, et `ESCAPED BY` sont aussi optionnelles, mais vous devez en spècifier au moins une.

Si vous ne spècifiez par de clause `FIELDS`, les valeurs par dèfaut sont :

```
FIELDS TERMINATED BY '\t' ENCLOSED BY '' ESCAPED BY '\\'
```

Si vous ne spècifiez par de clause `LINES`, les valeurs par dèfaut sont :

```
LINES TERMINATED BY '\n'
```

En d'autres termes, les valeurs par dèfaut font que `LOAD DATA INFILE` lit les donnèes comme suit :

- Recherche des limites de lignes parmis les nouvelles lignes.
- Scinde les lignes en champs avec les tabulations.
- Ne suppose pas que les champs sont entourès de guillemets.
- Interpréte les occurrences de tabulation, nouvelle ligne, '`\`' prècèdèes par '`\`' comme des caractéres littèraux qui font partie de la valeur d'un champs.

A l'inverse, les valeurs par dèfaut font que `SELECT ... INTO OUTFILE` ècrit les donnèes comme ceci :

- Ecrit des tabulations entre les champs.
- N'entoure pas les champs de guillemets.

- Utilise '\' pour echapper les occurences de tabulation, nouvelle ligne, '\' trouvèes dans les valeurs.
- Insére une nouvelle ligne entre les lignes.

Notez que pour utiliser `FIELDS ESCAPED BY '\\'`, vous devez spècifier deux antislashs pour que cette valeur soit interprètèe comme un antislash simple.

L'option `IGNORE nombre LINES` sert á ignorer une en-tête de fichier, telle que des noms de colonnes, qui dèbutent parfois un fichier á charger :

```
mysql> LOAD DATA INFILE "/tmp/nom_fichier" INTO TABLE test IGNORE 1 LINES;
```

Lorsque vous utilisez `SELECT ... INTO OUTFILE` conjointement avec `LOAD DATA INFILE` pour ècrire des donnèes dans un fichier et les relire dans une table, les options de `FIELDS` et `LINES` doivent être identiques. Sinon, `LOAD DATA INFILE` ne pourra pas interprèter le contenu du fichier correctement. Supposez que la commande `SELECT ... INTO OUTFILE` ait ècrit un fichier dèlimitè par des virgules :

```
mysql> SELECT * INTO OUTFILE 'donnees.txt'
    ->          FIELDS TERMINATED BY ','
    ->          FROM ...;
```

Pour lire ce fichier, la commande correcte serait :

```
mysql> LOAD DATA INFILE 'donnees.txt' INTO TABLE table2
    ->           FIELDS TERMINATED BY ',';
```

Si au contraire, vous essayez de lire le fichier avec la commande ci-dessous, cela ne fonctionnera pas, car la commande `LOAD DATA INFILE` essaie de lire des tabulations entre les champs :

```
mysql> LOAD DATA INFILE 'donnees.txt' INTO TABLE table2
    ->           FIELDS TERMINATED BY '\t';
```

Il est probable que chaque ligne d'entrèe sera interprètèe que comme un seul champ.

La commande `LOAD DATA INFILE` peut être utilisèe pour lire des donnèes issues d'autres sources. Par exemple, un fichier au format dBASE prèsente des champs sèparès par des virgules, et entourès de guillemets doubles. Si les lignes sont terminèes par de nouvelles lignes, la commande ci-dessous illustre la relecture d'un tel fichier avec MySQL :

```
mysql> LOAD DATA INFILE 'donnees.txt' INTO TABLE nom_de_table
    ->           FIELDS TERMINATED BY ',' ENCLOSED BY '"'
    ->           LINES TERMINATED BY '\n';
```

Les clauses `FIELDS` et `LINES` peuvent prendre des chaînes vides comme valeur. S'il la chaîne n'est pas vide, `FIELDS [OPTIONALLY] ENCLOSED BY` et `FIELDS ESCAPED BY` ne doivent avoir qu'un seul caractére. Les valeurs de `FIELDS TERMINATED BY` et `LINES TERMINATED BY` peuvent avoir plus d'un caractére. Par exemple, pour ècrire des lignes terminèes par le couple retour chariot-nouvelle ligne, ou pour lire un tel fichier, spècifiez la clause `LINES TERMINATED BY '\r\n'`.

Par exemple, pour charger un fichier de blagues, qui sont sèparèes par une ligne de `%%`, dans une table vous pouvez faire :

```
CREATE TABLE blagues (
   a INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
   blague TEXT NOT NULL
```

```
   );
LOAD DATA INFILE "/tmp/blagues.txt" INTO TABLE blagues FIELDS TERMINATED BY ""
LINES TERMINATED BY "\n%%\n" (blague);
```

`FIELDS [OPTIONALLY] ENCLOSED BY` contrôle la mise entre guillemets des champs. Pour l'ècriture de fichier (`SELECT ... INTO OUTFILE`), si vous omettez le mot `OPTIONALLY`, tous les champs seront entourès par le caractére spècifiè dans la clause `ENCLOSED BY`. Par exemple, si la virgule est utilisèe comme sèparateur de champs :

```
"1","une chaîne","100.20"
"2","une chaîne contenant une , virgule","102.20"
"3","une chaîne contenant un \" guillemet","102.20"
"4","une chaîne contenant un \", guillemet et une virgule","102.20"
```

Si vous spècifiez `OPTIONALLY`, le caractére `ENCLOSED BY` n'est utilisè que pour protèger les colonnes de types `CHAR` et `VARCHAR` :

```
1,"une chaîne",100.20
2,"une chaîne contenant une , virgule",102.20
3,"une chaîne contenant un \" guillemet",102.20
4,"une chaîne contenant un \", guillemet et une virgule",102.20
```

Notez que les occurrences du caractére `ENCLOSED BY` dans un champs sont èchappèe en les prèfixant avec le caractére `ESCAPED BY`. Notez aussi que si vous spècifiez un caractére d'èchappement vide, il n'est pas possible de garantir que les champs seront correctement relus par `LOAD DATA INFILE`. Par exemple, l'exemple ci-dessus apparaîtra comme montrè ci-dessous. Notez que le second champ de la quatriéme ligne comporte une virgule suivant un guillemet qui semble (mais c'est faux) terminer la ligne :

```
1,"une chaîne",100.20
2,"une chaîne contenant une , virgule",102.20
3,"une chaîne contenant un " guillemet",102.20
4,"une chaîne contenant un ", guillemet et une virgule",102.20
```

Lors des lectures, le caractére `ENCLOSED BY`, s'il est prèsent, est supprimè des extrèmitès de la valeur du champ. (ce qui est vrai, qu'il y ait l'option `OPTIONALLY` ou pas). Les occurences du caractére `ENCLOSED BY`, prècèdèes par le caractére `ESCAPED BY` sont interprètèes comme faisant partie de la valeur du champ. Les caractéres `ENCLOSED BY` doublèes, apparaissant dans la chaîne, sont interprètès comme le caractére `ENCLOSED BY` lui-même. Par exemple, si `ENCLOSED BY '"'` est spècifiè, les guillemets sont gèrès comme ceci :

```
"Le ""GRAND"" chef"  -> Le "GRAND" chef
Le "GRAND" chef      -> Le "GRAND" chef
Le ""GRAND"" chef    -> Le ""GRAND"" chef
```

`FIELDS ESCAPED BY` contrôle les caractéres spèciaux. Si le caractére `FIELDS ESCAPED BY` n'est pas vide, il est utilisè pour prèfixer les caractéres suivants en ècriture :

- La caractére `FIELDS ESCAPED BY`
- Le caractére `FIELDS [OPTIONALLY] ENCLOSED BY`
- Le premier caractére des valeurs de `FIELDS TERMINATED BY` et `LINES TERMINATED BY`
- ASCII `0` (en fait, ce qui est ècrit aprés le caractére d'èchappement est le caractére ASCII `'0'`, et non pas le code ASCI de zèro)

Si le caractére `FIELDS ESCAPED BY` est vide, aucun caractére ne sera èchappè. Ce n'est probablement pas une bonne idèe de spècifier un caractére d'èchappement vide, en particulier si les valeurs dans vos champs risquent d'utiliser l'un des caractéres de la liste ci-dessus.

En lecture, si le caractére `FIELDS ESCAPED BY` n'est pas vide, les occurrences de ce caractére sont supprimèes, et le caractére suivant est lu littèralement. Les exceptions á cette régle sont '`0`' ou '`N`' (par exemple, `0` ou `\N` si le caractére d'èchappement est '`\`'). Ces sèquences sont interprètèes comme l'octet nul (ASCII `0`) et la valeur `NULL`. Voyez plus bas pour la gestion des valeurs `NULL`.

Pour plus d'informations sur la syntaxe avec les caractéres d'èchappement '`\`', consultez Section 6.1.1 [Literals], page 404.

Dans certains cas, les options de `FIELDS` et `LINES` interférent entre elles :

- Si le caractére de `LINES TERMINATED BY` est une chaîne vide et que celui de `FIELDS TERMINATED BY` ne l'est pas, ce dernier sera celui utilisè pour `LINES TERMINATED BY`.
- Si les valeurs `FIELDS TERMINATED BY` et `FIELDS ENCLOSED BY` sont vides toutes les deux (`''`), un format á taille de champ fixe est utilisè. Avec ce format, aucun dèlimiteur n'est utilisè entre les champs. Au lieu de cela, les valeurs des colonnes sont ècrites avec leur configuration d'affichage. Par exemple, si une colonne a ètè dèclarèe `INT(7)`, la valeur de cette colonne sera ècrite avec 7 caractéres. Lors de la relecture, la valeur de la colonne sera obtenue en lisant á nouveau 7 caractéres. Ce format á taille fixe affecte la gestion de la valeur `NULL`; voyez plus loin pour cela. Notez que ce format ne fonctionne pas avec les jeux de caractéres multi-octets.

La gestion des valeurs `NULL` depend des options `FIELDS` et `LINES` que vous utilisez :

- Pour les valeurs par dèfaut de `FIELDS` et `LINES`, `NULL` est ècrit `\N` et `\N` est lu `NULL` (en supposant que le caractére d'èchappement est '`\`').
- Si `FIELDS ENCLOSED BY` n'est pas vide, un champ contenant le mot `NULL` comme valeur sera lu comme la valeur `NULL` (ce qui différe du mot `NULL`, entourè du caractére `FIELDS ENCLOSED BY`, qui sera lu comme le mot `'NULL'`).
- Si `FIELDS ESCAPED BY` est vide, `NULL` est ècrit comme le mot `'NULL'`.
- Avec le format á taille fixe (ce qui arrive si `FIELDS TERMINATED BY` et `FIELDS ENCLOSED BY` sont tous les deux vides), les valeurs `NULL` sont ècrites sous forme de chaîne vide. Notez que cela fait que `NULL` et les chaînes vides seront reprèsentèes par une valeur qui ne les distingue pas l'une de l'autre. Si vous avez besoin de diffèrencier entre les deux, n'utilisez par ce format !

Certains cas ne sont pas supportès par `LOAD DATA INFILE`:

- Lignes á tailles fixes (`FIELDS TERMINATED BY` et `FIELDS ENCLOSED BY` sont tous les deux vides) et les types de colonne `BLOB` ou `TEXT`.
- Si vous spècifiez un sèparateur qui est le même qu'un autre prèfixe, `LOAD DATA INFILE` ne sera pas capable de relire proprement le rèsultat. Par exemple, la clause `FIELDS` suivante posera sûrement des problémes :

```
FIELDS TERMINATED BY '"' ENCLOSED BY '"'
```

- Si `FIELDS ESCAPED BY` est vide, une valeur de colonne qui contient une occurence de `FIELDS ENCLOSED BY` ou de `LINES TERMINATED BY` suivi du caractére `FIELDS TERMINATED BY` interrompra la lecture de `LOAD DATA INFILE` trop tôt. Cela est dû

au fait que `LOAD DATA INFILE` ne peut pas faire la diffèrence entre la valeur dans le champ et la fin de la ligne.

L'exemple suivant charge toutes les colonnes de la table `persondata` :

```
mysql> LOAD DATA INFILE 'persondata.txt' INTO TABLE persondata;
```

Aucun champ n'est spècifiè, ce qui fait que `LOAD DATA INFILE` s'attend á ce que les lignes lues contiennent le bon nombre de champs. Les valeurs par dèfaut de `FIELDS` et `LINES` sont utilisèes.

Si vous voulez charger uniquement quelques colonnes dans une table, spècifiez la liste des champs :

```
mysql> LOAD DATA INFILE 'persondata.txt'
    ->           INTO TABLE persondata (col1,col2,...);
```

Vous devez aussi spècifier les champs si l'ordre dans lequel ils seront lus différe de l'ordre des colonnes dans la table. Sinon, MySQL ne pourra pas savoir á quelle colonne correspond une valeur.

Si une ligne a trop peu de champs, les colonnes omises prendront leur valeur par dèfaut. Les affectations de valeurs par dèfaut sont dècrites dans Section 6.5.3 [`CREATE TABLE`], page 504.

Une valeur de champs vide et un champ manquant ne seront pas interprètès de la même façon :

- Pour les types chaîne, la colonne est remplie avec la chaîne vide.
- Pour les types numèriques, la colonne est mise á `0`.
- Pour les types dates et heures, la colonne est mise au zèro appropriè pour le type. Voir Section 6.2.2 [Date and time types], page 423.

Notez que vous obtiendrez le même rèsultat en assignant á ces diffèrents types de champs la chaîne vide dans une commande `INSERT` ou `UPDATE`.

Les colonnes `TIMESTAMP` prendront la date et l'heure courante uniquement si on leur affecte la valeur NULL, ou (pour la premiére colonne `TIMESTAMP` seulement) si la colonne `TIMESTAMP` est ignorèe de la liste des colonnes spècifièe.

Si une ligne d'entrèe comporte trop de colonnes, les champs en trop sont ignorès, et le nombre d'alertes est incrèmentè.

`LOAD DATA INFILE` considére toutes les valeurs lues comme des chaînes de caractéres : vous ne pourrez donc pas utiliser la forme numèrique des colonnes `ENUM` ou `SET`, comme d'habitude. Toutes les colonnes `ENUM` et `SET` doivent être spècifièe comme des chaînes !

`LOAD DATA INFILE` regards all input as strings, so you can't use numeric values for `ENUM` or `SET` columns the way you can with `INSERT` statements. All `ENUM` and `SET` values must be specified as strings!

Si vous utilisez l'API C, vous pouvez obtenir des informations á propos de la requête en utilisant la fonction `mysql_info()` quand `LOAD DATA INFILE` se termine. Le format de la chaîne d'informations est le suivant :

```
Records: 1  Deleted: 0  Skipped: 0  Warnings: 0
```

Les alertes sont gènèrèes dans les mêmes circonstances que pour la commande `INSERT` (voir Section 6.4.3 [`INSERT`], page 489), exceptè que `LOAD DATA INFILE` gènére aussi des alertes s'il y a trop peu ou trop de champs dans une ligne. Les alertes ne sont pas stockèes; le

nombre d'alertes est la seule indication. Si vous recevez des alertes et vous voulez savoir exactement ce qui s'est passè, exècutez une commande `SELECT ... INTO OUTFILE` dans un autre fichier et comparez le avec le fichier original.

Si vous voulez faire lire `LOAD DATA` á partir d'un pipe, vous pouvez utiliser l'astuce suivante :

```
mkfifo /mysql/db/x/x
chmod 666 /mysql/db/x/x
cat < /dev/tcp/10.1.1.12/4711 > /nt/mysql/db/x/x
mysql -e "LOAD DATA INFILE 'x' INTO TABLE x" x
```

Si vous utilisez une version de MySQL supèrieure á 3.23.25 vous ne pouvez que faire ce qui prècéde avec `LOAD DATA LOCAL INFILE`.

Pour plus d'informations sur les performances de `INSERT` comparèes á `LOAD DATA INFILE` et accèlerer `LOAD DATA INFILE` : Voir Section 5.2.9 [Insert speed], page 374.

6.4.10 Syntaxe de `DO`

```
DO expression, [expression, ...]
```

Exècute l'expression mais ne retourne aucun rèsultat. C'est un alias de `SELECT expression, expression`, mais il a l'avantage d'être plus rapide quand on n'a pas besoin du rèsultat.

Cela s'avére trés utile avec les fonctions qui ont des effets secondaires, comme `RELEASE_LOCK`.

6.5 Dèfinition de donnèes : `CREATE`, `DROP`, `ALTER`

6.5.1 Syntaxe de `CREATE DATABASE`

```
CREATE DATABASE [IF NOT EXISTS] db_name
```

`CREATE DATABASE` crèe une base de donnèes avec le nom donnè. La syntaxe des noms des bases de donnèes est donnèe ici : Section 6.1.2 [Legal names], page 407. Une erreur survient si la base de donnèes existe dèjá et que vous n'avez pas spècifiè `IF NOT EXISTS`.

Les bases de donnèes MySQL sont implèmentèes comme des rèpertoires contenant des fichiers qui correspondent aux tables dans les bases de donnèes. Puisqu'il n'y a pas de tables dans une base de donnèes lors de sa crèation, la requête `CREATE DATABASE` crèera seulement le dossier dans le rèpertoire de donnèes de MySQL.

Vous pouvez aussi crèer des bases de donnèes avec `mysqladmin`. Voir Section 4.8 [Client-Side Scripts], page 305.

6.5.2 Syntaxe de `DROP DATABASE`

```
DROP DATABASE [IF EXISTS] db_name
```

`DROP DATABASE` dètruit toutes les tables dans la base de donnèes et l'efface elle même. Si vous utilisez la commande `DROP DATABASE` sur un lien symbolique pointant sur la base de donnèes, le lien et la base seront effacès. **Soyez trés prudents avec cette commande!**

`DROP DATABASE` retourne le nombre de fichiers qui ont ètè effacès du dossier de la base de donnèes. Normalement, c'est ègal á trois fois le nombre de tables, car chaque table, correspond normalement á un fichier '`.MYD`', un fichier '`.MYI`' et un fichier '`.frm`'.

La commande `DROP DATABASE` efface tous les fichiers du dossier de la base de donnèes ayant les extensions suivantes :

Ext	**Ext**	**Ext**	**Ext**
.BAK	.DAT	.HSH	.ISD
.ISM	.ISM	.MRG	.MYD
.MYI	.db	.frm	

Tous les sous-dossiers qui consistent de 2 chiffres (dossiers `RAID`) sont aussi supprimès.

Depuis la version 3.22 de MySQL, vous pouvez utiliser le mot clef `IF EXISTS` pour èviter l'affichage d'erreurs si la base n'existe pas.

Vous pouvez aussi supprimer des bases de donnèes avec `mysqladmin`. Voir Section 4.8 [Client-Side Scripts], page 305.

6.5.3 Syntaxe de `CREATE TABLE`

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] nom_de_table [(definition_de_create,...)]
[options_de_table] [select_statement]

definition_de_create:
  nom_de_colonne type [NOT NULL | NULL] [DEFAULT valeur_par_defaut] [AUTO_INCREMENT
            [PRIMARY KEY] [definition_de_reference]
  ou    PRIMARY KEY (index_col_name,...)
  ou    KEY [nom_index] (index_col_name,...)
  ou    INDEX [nom_index] (index_col_name,...)
  ou    UNIQUE [INDEX] [nom_index] (index_col_name,...)
  ou    FULLTEXT [INDEX] [nom_index] (index_col_name,...)
  ou    [CONSTRAINT symbol] FOREIGN KEY [nom_index] (index_col_name,...)
            [reference_definition]
  ou    CHECK (expr)

type:
        TINYINT[(longueur)] [UNSIGNED] [ZEROFILL]
  ou    SMALLINT[(longueur)] [UNSIGNED] [ZEROFILL]
  ou    MEDIUMINT[(longueur)] [UNSIGNED] [ZEROFILL]
  ou    INT[(longueur)] [UNSIGNED] [ZEROFILL]
  ou    INTEGER[(longueur)] [UNSIGNED] [ZEROFILL]
  ou    BIGINT[(longueur)] [UNSIGNED] [ZEROFILL]
  ou    REAL[(longueur,dècimales)] [UNSIGNED] [ZEROFILL]
  ou    DOUBLE[(longueur,dècimales)] [UNSIGNED] [ZEROFILL]
  ou    FLOAT[(longueur,dècimales)] [UNSIGNED] [ZEROFILL]
  ou    DECIMAL(longueur,dècimales) [UNSIGNED] [ZEROFILL]
  ou    NUMERIC(longueur,dècimales) [UNSIGNED] [ZEROFILL]
  ou    CHAR(longueur) [BINARY]
  ou    VARCHAR(longueur) [BINARY]
```

```
  ou    DATE
  ou    TIME
  ou    TIMESTAMP
  ou    DATETIME
  ou    TINYBLOB
  ou    BLOB
  ou    MEDIUMBLOB
  ou    LONGBLOB
  ou    TINYTEXT
  ou    TEXT
  ou    MEDIUMTEXT
  ou    LONGTEXT
  ou    ENUM(valeur1,valeur2,valeur3,...)
  ou    SET(valeur1,valeur2,valeur3,...)

index_nom_de_colonne:
        nom_de_colonne [(longueur)]

definition_de_reference:
        REFERENCES nom_de_table [(index_nom_de_colonne,...)]
                   [MATCH FULL | MATCH PARTIAL]
                   [ON DELETE option_de_reference]
                   [ON UPDATE option_de_reference]

option_de_reference:
        RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT

options_de_table:
TYPE = {BDB | HEAP | ISAM | InnoDB | MERGE | MRG_MYISAM | MYISAM }
ou AUTO_INCREMENT = #
ou AVG_ROW_LENGTH = #
ou CHECKSUM = {0 | 1}
ou COMMENT = "phrase"
ou MAX_ROWS = #
ou MIN_ROWS = #
ou PACK_KEYS = {0 | 1 | DEFAULT}
ou PASSWORD = "mot"
ou DELAY_KEY_WRITE = {0 | 1}
ou      ROW_FORMAT= { default | dynamic | fixed | compressed }
ou RAID_TYPE= {1 | STRIPED | RAID0 } RAID_CHUNKS=#  RAID_CHUNKSIZE=#
ou UNION = (nom_de_table,[nom_de_table...])
ou INSERT_METHOD= {NO | FIRST | LAST }
ou      DATA DIRECTORY="chemin absolu vers dossier"
ou      INDEX DIRECTORY="chemin absolu vers dossier"

select_statement:
[IGNORE | REPLACE] SELECT ...  (une clause de sèlèction valide)
```

`CREATE TABLE` Crèe une table avec le nom donnè, dans la base de donnèes courante. Les régles de nommage des tables sont disponibles dans Section 6.1.2 [Legal names], page 407. Une erreur est affichèe s'il n'y a pas de base courante, ou si la table existe dèjá.

Depuis la version 3.22 de MySQL, vous pouvez utiliser le mot rèservè `TEMPORARY` lorsque vous crèez une table. Une table temporaire sera immèdiatement effacèe dés que la connexion se termine. Cela signifie que vous pouvez utiliser le même nom de table temporaire depuis deux connexions diffèrentes sans risque de conflit entre les connexions. Vous pouvez aussi utiliser une table temporaire qui a le même nom qu'une table existante (la table existante est alors cachèe tant que dure la table temporaire). En MySQL version 4.0.2 ou plus rècent, vous avez juste á avoir le privilége `CREATE TEMPORARY TABLES` pour crèer des tables temporaires.

Depuis la version 3.23 de MySQL, vous pouvez utiliser le mot rèservè `IF NOT EXISTS`, de façon á ce qu'aucune erreur ne soit affichè si la table que vous essayez de crèer existe dèjá. Notez qu'il n'y a pas de comparaisons entre les structures de table lors du test d'existence.

Chaque table nom_de_table est representèe par des fichiers dans le dossier de la base de donnèes. Dans le cas des tables de type MyISAM, ce sont les fichiers suivants :

Fichier	**Rôle**
`nom_de_table.frm`	Fichier de dèfinition de la table
`nom_de_table.MYD`	Fichier de donnèes
`nom_de_table.MYI`	Fichier d'index

Pour plus d'informations sur les propriètès des diffèrentes colonnes et de leurs types, voyez Section 6.2 [Column types], page 416:

- Si ni `NULL`, ni `NOT NULL` n'est spècifiè, une colonne utilisera par dèfaut l'attribut `NULL` (elle acceptera les valeurs `NULL`).
- Une colonne de nombre entier peut se voir attribuer l'attribut `AUTO_INCREMENT`. Lorsque vous insèrez la valeur `NULL` (recommandèe) ou `0` dans une colonne `AUTO_INCREMENT`, la colonne prendra automatiquement la valeur de `value+1`, oú value est la plus grande valeur positive courante dans cette colonne. La sèrie des valeurs `AUTO_INCREMENT` commence á `1`. Voir Section 8.4.3.30 [`mysql_insert_id()`], page 635.

  Si vous effacez la ligne contenant la valeur maximale dans la colonne `AUTO_INCREMENT`, cette valeur sera rèutilisèe dans les tables de type ISAM mais pas dans les tables de type MyISAM. Si vous effacez toutes les lignes dans la table avec la commande `DELETE FROM` nom_de_table (sans la clause `WHERE)` en mode `AUTOCOMMIT`, la sèrie des valeurs `AUTO_INCREMENT` recommencera á zèro.

  NOTE : Il ne peut y avoir qu'une seule colonne de type `AUTO_INCREMENT` dans une table, et elle doit être indexèe. MySQL version 3.23 ne fonctionnera correctement que si cette colonne n'accueille que des valeurs positives. Insèrer un nombre nègatif sera considèrè comme inserer un nombre de trés grande taille, mais positif. Ceci est fait pour èviter les problémes de prècision lorsque les nombres passe de positif á nègatif lorsqu'ils atteignent leur valeur maximale positive. C'est aussi pour èviter qu'une colonne de type `AUTO_INCREMENT` ne contienne de valeur 0.

En MyISAM et tables BDB, vous pouvez spècifier une colonne secondaire d'`AUTO_INCREMENT` dans une clef multi-colonne.Voir Section 3.5.9 [example-AUTO_INCREMENT], page 184.

Pour rendre MySQL avec certaines applications ODBC, vous pouvez retrouver la valeur de la derniére valeur automatiquement gènèrèe avec la requête suivante :

```
SELECT * FROM nom_de_table WHERE auto_col IS NULL
```

- `CREATE TABLE` effectue automatiquement la transaction courante d'InnoDB si le MySQL binlogging est employè.
- La valeur `NULL` est traitèe diffèremment dans les colonnes de type `TIMESTAMP`. Vous ne pouvez pas stocker de valeur `NULL` littèrale dans une colonne `TIMESTAMP`; insèrer une valeur `NULL` dans une telle colonne revient á insèrer la date et l'heure courante. Car les colonnes `TIMESTAMP` ignorent les attributs NULL et `NOT NULL`.

  Cela facilite grandement l'utilisation des colonnes `TIMESTAMP` pour les clients MySQL : le serveur indique que ces colonnes peuvent se voir assigner une valeur `NULL` (ce qui est vrai), même si les colonnes `TIMESTAMP` ne contiendront jamais de valeur `NULL`. Vous pouvez le constater lorsque vous utiliser la commande `DESCRIBE` nom_de_table pôur avoir une description de votre table.

  Notez qu'affecter la valeur `0` á une colonne `TIMESTAMP` n'est pas la même chose que lui affecter la valeur `NULL`, car `0` est une valeur `TIMESTAMP` valide.
- Une valeur `DEFAULT` doit être une constante, ça ne peut être une fonction ou une expression.

  Si aucun valeur par dèfaut (attribut `DEFAULT`) n'est spècifiè, MySQL en assigne une automatiquement

  Si la colonne accepte les valeur `NULL`, la valeur par dèfaut sera la valeur `NULL`.

  Si la colonne est dèclarèe comme `NOT NULL` (non-nulle), la valeur par dèfaut dèpendra du type de colonne :

  – Pour les types numèriques sans l'attribut `AUTO_INCREMENT`, la valeur sera `0`. Pour une colonne `AUTO_INCREMENT`, la valeur par dèfaut sera la prochaine valeur de la sèrie.

  – Pour les types dates et heures autres que `TIMESTAMP`, la valeur par dèfaut est la date zèro approprièe. Pour les colonnes `TIMESTAMP`, la valeur par dèfaut est la date et l'heure courante. Voir Section 6.2.2 [Date and time types], page 423.

  – `DEFAULT` directive). Pour les chaînes autres que `ENUM`, la valeur par dèfaut est la chaîne vide. Pour les valeurs de type `ENUM`, la valeur par dèfaut est le premier èlèment de l'ènumèration.
- `KEY` est un synonyme de `INDEX`.
- Les valeurs par dèfaut doivent être des constantes. Cela signifie, par exemple, que vous ne pouvez pas donner de valeur par dèfaut en fonction de `NOW()` ou `CURRENT_DATE`.
- Une clè primaire (`PRIMARY KEY`) est un index `UNIQUE` avec la contrainte supplèmentaire que les toutes les colonnes utilisèes doit avoir l'attribut `NOT NULL`. En MySQL, cette clè est dite `PRIMARY`. Une table ne peut avoir qu'une seule clè primaire. Si vous n'avez pas de `PRIMARY KEY` et que des applications demandent la `PRIMARY KEY` dans vos tables, MySQL retournera la premiére clè `UNIQUE`, qui n'a aucune valeur `NULL`.

- Une `PRIMARY KEY` peut être multi-colonne. Cependant, vous ne pouvez pas crèer d'index multi-colonne avec l'attribut `PRIMARY KEY` dans une spècification de colonne. En faisant cela, le seul rèsultat sera que cette seule colonne sera marquèe comme clè primaire. Vous devez absolument utiliser la syntaxe `PRIMARY KEY` (index_nom_de_colonne, ...).
- Si une clè primaire (`PRIMARY`) ou unique (`UNIQUE`) est ètablit sur une seule colonne, et que cette colonne est de type entier, vous pouvez aussi faire rèfèrence á cette colonne sous le nom _rowid (nouveau en version 3.23.11).
- Si vous ne donnez pas de nom á un index, l'index prendra le nom de la premiére colonne qui le compose, avec èventuellement un suffixe (_2, _3, ...) pour le rendre unique. Vous pouvez voir les noms des index avec la commande `SHOW INDEX FROM` nom_de_table. Voir Section 4.5.6 [`SHOW`], page 267.
- Seul, les formats de table `MyISAM`, `InnoDB`, et `BDB` supportent des index sur des colonnes qui peuvent contenir des valeurs `NULL`. Dans les autres situations, vous devez dèclarer ces colonnes `NOT NULL` ou une erreur sera gènèrèe.
- Avec la syntaxe nom_de_colonne(longueur), vous pouvez spècifier un index qui n'utilise qu'une partie de la colonne `CHAR` ou `VARCHAR`. Cela peut rèduire la taille des fichiers d'index. Voir Section 5.4.4 [Indexes], page 386.
- Seul le format de table `MyISAM` supporte l'indexation des colonnes `BLOB` et `TEXT`. Colonnes `TEXT`. Lorsque vous ajoutez un index á une colonne `BLOB` ou `TEXT`, vous devez ABSOLUMENT spècifier une longueur d'index :

```
CREATE TABLE test (blob_col BLOB, INDEX(blob_col(10)));
```

- Lorsque vous utilisez une clause `ORDER BY` ou `GROUP BY` sur une colonne de type `TEXT` ou `BLOB`, seuls, les max_sort_longueur premiers octets seront lus. Voir Section 6.2.3.2 [`BLOB`], page 430.
- En MySQL version 3.23.23 ou plus rècent, vous pouvez aussi crèer des index spècial `FULLTEXT`. Ils sont utilisès pour faire des recherches en texte plein. Seul, le format de table `MyISAM` supporte les index `FULLTEXT`. Ils peuvent être crèès uniquement pour les colonnes de type `VARCHAR` et `TEXT`. L'indexation est alors exècutèe sur toute la colonne, et les indexations partielles ne sont pas supportèes. supported. Voir Section 6.8 [Fulltext Search], page 522 pour les dètails.
- Chaque colonne `NULL` requiert un bit supplèmentaire, arrondi á l'octet supèrieur le plus proche.
- La taille maximale d'enregistrement peut être calculèe comme ceci :

```
row longueur = 1
             + (somme des longueurs de colonnes)
             + (nombre de colonnes NULL + 7)/8
             + (nombre de colonnes á taille variable)
```

- Les options options_de_table et `SELECT` ne sont implèmentèes que dans MySQL version 3.23 et plus rècent.

  Les diffèrents types de tables sont :

Table type	**Description**
BDB or BerkeleyDB	Tables avec transactions. Voir Section 7.6 [BDB], page 585.
HEAP	Les donnèes de ces tables ne sont stockèes qu'en mèmoire. Voir Section 7.4 [HEAP], page 543.

ISAM	Le gestionnaire originel de tables. Voir Section 7.3 [ISAM], page 542.
InnoDB	Transaction-safe tables with row locking. Voir Section 7.5 [InnoDB], page 544.
MERGE	Un emsemble de tables MyISAM utilisèes comme une seule et même table. Voir Section 7.2 [MERGE], page 539.
MRG_MyISAM	Un synonyme pour MERGE les tables.
MyISAM	Le nouveau gestionnaire de table binaire et portable. Voir Section 7.1 [MyISAM], page 532.

Voir Chapitre 7 [Table types], page 531.

Si un type de table est demandè, mais que ce type particulier n'est pas disponible, MySQL va choisir le type de table le plus proche de celui qui est spècifiè. Par exemple, si `TYPE=BDB` est spècifiè, et que la distribution de MySQL ne supporte pas les tables `BDB`, la table qui sera crèèe sera du type `MyISAM`.

Les autres options de tables sont utilisèes pour optimiser le comportement de la table. Dans la plupart des cas, vous n'avez pas á les spècifier. Les options fonctionnent pour tous les types de tables (sauf contre-indication) :

Option	**Description**
`AUTO_INCREMENT`	La prochaine valeur auto_increment de votre table (MyISAM)..
`AVG_ROW_LENGTH`	La taille moyenne approchèe des lignes de votre table. Vous ne devez fournir cette valeur que pour les tables á taille de ligne variable, de trés grande taille.
`CHECKSUM`	Passez 1 si vous voulez que MySQL gènére une somme de vèrification (ce qui facilite la recherche des lignes corrompues, mais ralentit les mises á jour) (MyISAM).
`COMMENT`	Un commentaire pour votre table (60 caractéres).
`MAX_ROWS`	Nombre de lignes maximum que vous pensez stocker dans la table.
`MIN_ROWS`	Nombre de minimum lignes que vous pensez stocker dans la table.
`PACK_KEYS`	Passez 1 si vous voulez un index plus compact. Cela rend les mises á jour plus lente, mais les lectures plus rapides (MyISAM, ISAM).
`PASSWORD`	Chiffre le fichier .frm avec un mot de passe. Cette option ne fait rien du tout pour la version standard de MySQL.
`DELAY_KEY_WRITE`	Passez 1 si vous voulez attendre la fermeture de la table pour mettre á jour les index.
`ROW_FORMAT`	Definit la mèthode de stockage des lignes (rèservè pour le futur). Actuellement, cette option fonctionne uniquement avec des tables MySAM qui supportent le `DYNAMIC` et `FIXED` en format de ligne. Voir Section 7.1.2 [MyISAM table formats], page 535.

Lorsque vous utilisez une table MyISAM, MySQL utilise le produit `max_rows * avg_row_longueur` pour dècider de la taille de la table. Si vous ne spècifiez pas ces options, la taille maximum sera 4Go (ou 2Go si votre systéme d'exploitation ne supporte que les tables de 2 Go). La raison de cette option est le choix des tailles de pointeurs d'index : plus la table sera petite, plus les index seront petits, et rapides á lire.

Si vous n'utilisez pas l'option `PACK_KEYS`, l'option par dèfaut est de ne compacter que les chaînes, et pas les nombres. Si vous passez `PACK_KEYS=1`, les nombres seront aussi compactès.

Lorsque vous compactez des clès binaires numèriques, MySQL utilisera la compression par prèfixe. Cela signifie que vous n'y aurez vraiment intèrêt, que si beaucoup de nombres sont identiques. La compression par prèfixe utilise un octet de plus pour chaque clè, pour indiquer le nombre d'octets de la clè courante, identique á la clè prècèdente (notez que le pointeur de ligne est stockè au format bigendian (les premiers bits ont le plus de poids), pour amèliorer la taux de compression). Cela signifie que si vous avez plusieurs clès de la même valeurs sur des lignes consècutives, les clès ne prendront que 2 octets (y compris le pointeur de ligne). Faites vous-même la comparaison avec la mèthode standard, oú la clè suivante occupe `storage_size_for_key + pointer_size` (gènèralement 4). D'un autre cotè, si toutes vos clès sont totalement diffèrente, vous perdre un autre octet par clè (si la clè ne peut avoir de valeur `NULL` : dans ce cas, la taille de la clè compactèe sera stockèe dans le même octet qui indique si la clè est `NULL`.)

- Si vous spècifiez une clause `SELECT` dans une commande `CREATE STATEMENT`, MySQL crèera de nouveaux champs pour tous les èlèments du `SELECT`. Par exemple:

```
mysql> CREATE TABLE test (a INT NOT NULL AUTO_INCREMENT,
    ->        PRIMARY KEY (a), KEY(b))
    ->        TYPE=MyISAM SELECT b,c FROM test2;
```

Cette ligne va crèer une table `MySAM` de 3 colonnes. Notez que cette table sera automatiquement supprimèe si une erreur survient durant la copie des donnèes dans la table. Voyez cet exemple:

```
mysql> SELECT * FROM foo;
+---+
| n |
+---+
| 1 |
+---+

mysql> CREATE TABLE bar (m INT) SELECT n FROM foo;
Query OK, 1 row affected (0.02 sec)
Records: 1  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM bar;
+------+---+
| m    | n |
+------+---+
| NULL | 1 |
+------+---+
1 row in set (0.00 sec)
```

Pour chaque ligne dans la table `foo`, une ligne est insèrèe dans `bar` avec la valeur de `foo` et les valeurs par dèfaut pour les nouvelles colonnes.

`CREATE TABLE ... SELECT` ne crèera pas d'index automatiquement pour vous. Cela est intentionnel, pour rendre la commande aussi flexible que possible. Si vous voulez avoir des index dans la table crèèe, vous devez le spècifier avant la commande `SELECT` :

```
mysql> CREATE TABLE bar (UNIQUE (n)) SELECT n FROM foo;
```

Si une erreur survient durant la copie des donnèes dans la table, elle sera automatiquement effacèe.

Pour d'assurer que les journaux des modifications ou les journaux binaires puissent être utilisès pour re-crèer les tables originales, MySQL n'autorise pas les insertions concurrents pendant `CREATE TABLE ... SELECT`.

- L'option `RAID_TYPE` vous aidera á passer outre la limite de 2Go/4Go pour les fichiers MyISAM (mais pas le fichier d'index), sur les systémes d'exploitation qui ne supportent pas les grands fichiers.

  Il est possible d'accèlèrer le goulet d'ètranglement des I/O en disposant les rèpertoires `RAID` sur diffèrents disques physiques. `RAID_TYPE` fonctionne sur tous les OS, á condition d'avoir configurè MySQL avec `--with-raid`. Actuellement, le seul `RAID_TYPE` autorisè est `STRIPED` (`1` et `RAID0` sont des redirections vers celui ci).

  Si vous spècifiez `RAID_TYPE=STRIPED` pour une table MyISAM, MyISAM va crèer des sous-dossiers `RAID_CHUNKS` nommès 00, 01, 02 dans le dossier de la base de donnèes. Dans chacun de ces dossiers, MyISAM va crèer un fichier nom_de_table.MYD. Lorsqu'il ècrira dans le fichier, le gestionnaire `RAID` placera les `RAID_CHUNKSIZE` *1024 premiers octets dans le premier fichier, et les `RAID_CHUNKSIZE` *1024 octets suivants dans le fichier suivant.
- `UNION` sert lorsque vous voulez utiliser un ensemble de tables comme une seule table. Cela ne fonctionne qu'avec les tables `MERGE`. Voir Section 7.2 [`MERGE`], page 539.

  Actuellement, vous devez avoir les droits de `SELECT`, `UPDATE`, et `DELETE` sur ces tables pour les consolider en une seule table `MERGE`. Toutes les tables doivent être de la même base que la table consolidèe.
- Si vous voulez insèrer des donnèes dans une table `MERGE`, vous devez spècifier avec `INSERT_METHOD`. Voir Section 7.2 [MERGE], page 539. Cette option a ètè introduite dans MySQL 4.0.0.
- Dans la table ainsi crèèe, la clè primaire `PRIMARY` sera placèe en premier, suivie des clès uniques et des clès standard. Cela aide l'optimiseur MySQL a utiliser les clès dans l'ordre de prioritè, et á dètecter les clès doubles.
- En utilisant `DATA DIRECTORY="directory"` ou `INDEX DIRECTORY="directory"` vous pouvez spècifier ou le gestionnaire de la table doit mettre la table et son index. Notez que le chemin doit être complet. Pas de chemin relatif.

  Cela fonctionne uniquement dans les tables `MyISAM` en `MySQL` 4.0, quand vous avez pas utilisè l'option `--skip-symlink`. Voir Section 5.6.1.2 [Symbolic links to tables], page 402.

6.5.3.1 Modification automatique du type de colonnes

Dans certains cas, MySQL change automatiquement la spècification d'une colonne fournie dans la commande `CREATE TABLE`. (Cela peut aussi arriver avec `ALTER TABLE`) :

- Les colonnes `VARCHAR` avec un taille infèrieure á quatre (4) sont changèes en `CHAR`.
- Si l'une des colonnes d'une table est de taille variable, toute la ligne est, par consèquent, de taille variable. Ansi, si une ligne contient une colonne de taille variable (`VARCHAR`, `TEXT` ou `BLOB`) toutes les colonnes `CHAR` de plus de trois caractéres sont transoformèes en `VARCHAR`. Cela ne change en rien la façon dont vous utilisez les colonnes. Pour MySQL, `VARCHAR` est simplement une autre façon de stocker les caractéres. MySQL effectue cette conversion car cela èconomise de la place, et rend les calculs sur les tables plus rapides. Voir Chapitre 7 [Table types], page 531.
- La taille d'affichage de `TIMESTAMP` doit être un nombre pair et être compris entre 2 et 14. (2, 4, 6, 8, 10, 12 ou 14). Si vous spècifiez une taille plus grande que 14, ou infèrieure á 2, celle-ci sera tranformèe en 14. Les valeurs impaires sont rammenèes á la valeur pair supèrieure la plus proche.
- # Vous ne pouvez pas stocker de valeur littèrale `NULL` dans une colonne de type `TIMESTAMP`. Cette valeur sera remplacèe par la date et l'heure courante. De ce fait, les attributs `NULL` et `NOT NULL` n'ont pas de sens pour ces colonnes et sont ignorès. `DESCRIBE nom_de_table` indiquera toujours que la colonne `TIMESTAMP` accepte les valeurs `NULL`.
- MySQL change certains type de colonnes utilisès par d'autres serveurs SQL en types MySQL. Voir Section 6.2.5 [Other-vendor column types], page 434.

Si vous voulez voir si MySQL a utilisè un autre type que celui que vous avez spècifiè, utilisez la commande `DESCRIBE nom_de_table`, aprés votre crèation ou modification de structure de table.

Certain types de colonnes peuvent être modifiès si vous compressez une table en utilisant l'utilitaire `myisampack`.

6.5.4 Syntaxe de `ALTER TABLE`

```
ALTER [IGNORE] TABLE nom_de_table alter_spec [, alter_spec ...]

alter_specification:
        ADD [COLUMN] create_definition [FIRST | AFTER nom_colonne ]
  ou    ADD [COLUMN] (create_definition, create_definition,...)
  ou    ADD INDEX [nom_index] (index_nom_colonne,...)
  ou    ADD PRIMARY KEY (index_nom_colonne,...)
  ou    ADD UNIQUE [nom_index] (index_nom_colonne,...)
  ou    ADD FULLTEXT [nom_index] (index_nom_colonne,...)
  ou ADD [CONSTRAINT symbol] FOREIGN KEY [nom_index] (nom_colonne_index,...)
           [reference_definition]
  ou    ALTER [COLUMN] nom_colonne {SET DEFAULT literal | DROP DEFAULT}
  ou    CHANGE [COLUMN] ancien_nom_colonne create_definition
                [FIRST | AFTER nom_de_colonne]
  ou    MODIFY [COLUMN] create_definition [FIRST | AFTER nom_colonne]
  ou    DROP [COLUMN] nom_colonne
  ou    DROP PRIMARY KEY
  ou    DROP INDEX nom_index
```

```
ou    DISABLE KEYS
ou    ENABLE KEYS
ou    RENAME [TO] nouveau_nom_de_table
ou    ORDER BY col
ou    table_options
```

`ALTER TABLE` vous permet de changer la structure d'une table existante. Par exemple, vous pouvez ajouter ou supprimer des colonnes, des index, changer le type des colonnes existantes, renommer ces colonnes, ou la table elle-même. Vous pouvez de même changer le commentaire sur la table, ou le type de celle-ci. Voir Section 6.5.3 [`CREATE TABLE`], page 504.

Si vous utilisez `ALTER TABLE` pour modifier les spècifications d'une colonne mais que `DESCRIBE nom_de_table` vous indique que cette colonne n'a pas ètè modifièe, il est possible que MySQL ait ignorè vos modifications pour une des raisons dècrite dans Section 6.5.3.1 [Silent column changes], page 511. Par exemple, si vous essayez de changer une colonne de type `VARCHAR` en `CHAR`, MySQL continuera d'utiliser `VARCHAR` si la table contient d'auttres colonnes de taille variable.

`ALTER TABLE` effectue une copie temporaire de la table originale. Les modifications sont faites sur cette copie, puis l'original est effacèe, et enfin la copie est renommèe pour remplacer l'originale. Cette mèthode permet de rediriger toutes les commandes automatiquement vers la nouvelle table sans pertes. Durant l'exècution de `ALTER TABLE`, la table originale est lisible par d'autres clients. Les modifications et insertions sont reportèes jusqu'á ce que la nouvelle table soit prête.

Notez que si vous utilisez une autre option que `RENAME` avec `ALTER TABLE`, MySQL crèera toujours une table temporaire, même si les donnèes n'ont pas besoin d'être copièes (comme quand vous changez le nom d'une colonne). Nous avons prèvu de corriger cela dans les versions suivantes, mais comme la commande `ALTER TABLE` n'est pas utilisèe trés souvent, cette correction ne fait pas partie de nos prioritès.

Pour les tables MyISAM, vous pouvez accèlèrer la recrèation de l'index (qui est la plus lente de tout le processus) en choisissant une valeur èlevèe pour la variable `myisam_sort_buffer_size`.

- Pour utiliser `ALTER TABLE`, vous devez avoir les droits `ALTER`, `INSERT`, et `CREATE` sur la table.
- `IGNORE` est une extension MySQL pour ANSI SQL92. Cette option contrôle la façon dont `ALTER TABLE` fonctionne s'il y'a des duplications sur une clef unique de la nouvelle table. Si `IGNORE` n'est pas spècifièe, la copie est annulèe et la table originale est restaurèe. Si `IGNORE` est spècifièe, les lignes contenant les èlèments doublons de la table seront effacèes, hormis la premiére, qui sera conservèe.
- Vous pouvez effectuer plusieurs opèrations de `ADD`, `ALTER`, `DROP`, et `CHANGE` dans une même commande `ALTER TABLE`. C'est une extension de MySQL á la norme ANSI SQL92, qui n'autorise qu'une seule modification par commande `ALTER TABLE`.
- `CHANGE nom_colonne`, `DROP nom_colonne`, et `DROP INDEX` sont des extensions de MySQL á la norme ANSI SQL92.
- `MODIFY` est une extension Oracle á `ALTER TABLE`.
- Le mot optionnel `COLUMN` est purement de la fioriture et peut être ignorè.

- Si vous utilisez `ALTER TABLE nom_de_table RENAME TO nouveau_nom` sans autre option, MySQL va simplement renommer les fichiers qui correspondent á la table `nom_de_table`. Il n'y a pas de crèation de fichier temporaire. Voir Section 6.5.5 [`RENAME TABLE`], page 516.
- La dèfinition `create_definition` utilise la même syntaxe pour les clauses `ADD` et `CHANGE` que dans `CREATE TABLE`. Notez que cette syntaxe inclut le nom de la colonne, et pas seulement son type Voir Section 6.5.3 [`CREATE TABLE`], page 504.
- Vous pouvez renommer une colonne avec la syntaxe `CHANGE ancien_nom_de_colonne create_definition`. Pour cela, indiquez l'ancien nom de la colonne, puis le nouveau nom et son type courant. Par exemple, pour renommer une colonne de type `INTEGER`, de `a` en `b`, vous pouvez faire ceci :

  ```
  mysql> ALTER TABLE t1 CHANGE a b INTEGER;
  ```

  Si vous ne voulez changer que le type de la colonne, avec la clause `CHANGE` vous devrez redonner le nom de la colonne. Par exemple :

  ```
  mysql> ALTER TABLE t1 CHANGE b b BIGINT NOT NULL;
  ```

  Cependant, á partir de la version 3.22.16a de MySQL, vous pouvez aussi utiliser la clause `MODIFY` pour changer le type d'une colonne sans la renommer :

  ```
  mysql> ALTER TABLE t1 MODIFY b BIGINT NOT NULL;
  ```
- Si vous utilisez les clauses `CHANGE` ou `MODIFY` pour rèduire la taille d'une colonne qui comportait un index sur une partie de la colonne (par exemple, si vous aviez un index sur 10 caractéres d'une colonne de type `VARCHAR`), vous ne pouvez pas rendre la colonne plus petite que le nombre de caractéres indexès.
- Quand vous changez le type d'une colonne avec `CHANGE` ou `MODIFY`, MySQL essaye de convertir les donnèes au niveau type dans la mesure du possible.
- A partir de la version 3.22 de MySQL, vous pouvez utiliser `FIRST` ou `ADD ... AFTER nom_colonne` pour ajouter la colonne á un endroit spècifique dans la table. Par dèfaut, la colonne est ajoutèe á la fin. A partir de la version 4.0.1, vous pouvez aussi utiliser les mots clès `FIRST` et `AFTER` avec `CHANGE` ou `MODIFY`.
- `ALTER COLUMN` spècifie une nouvelle valeur par dèfaut pour une colonne ou enléve l'ancienne. si l'ancienne valeur est effacèe et que la colonne peut être `NULL`, la nouvelle valeur par dèfaut sera `NULL`. Si la colonne ne peut être `NULL`, MySQL assigne une valeur par dèfaut, comme dècit dans Section 6.5.3 [`CREATE TABLE`], page 504.
- `DROP INDEX` supprime un index. C'est une extension MySQL á la norme ANSI SQL92. Voir Section 6.5.8 [DROP INDEX], page 517.
- Si des colonnes sont effacèes d'une table, ces colonnes sont aussi supprimès des index dont elles font partie. Si toutes les colonnes qui forment un index sont effacèes, l'index lui même est supprimè.
- Si une table ne comporte qu'une seule colonne, La colonne ne peut être supprimèe. Si vous voulez effacer la table, utilisez la commande `DROP TABLE`.
- `DROP PRIMARY KEY` supprime la clef primaire. Si cette clef n'existe pas, cette commande effacera le premier index `UNIQUE` de la table. (MySQL marque la premiére clef `UNIQUE` en tant que `PRIMARY KEY` si aucune `PRIMARY KEY` n'a ètè spècifièe explicitement.)

Si vous ajoutez un `UNIQUE INDEX` ou `PRIMARY KEY` á une table, c'est enregistrè avant les index non-`UNIQUE` pour que MySQL puisse dètecter les valeurs dupliquèes aussi vite que possible.

- `ORDER BY` vous permet de crèer une nouvelle table tout en ordonnant les lignes par dèfaut. Notez que cet ordre ne sera pas conservè aprés les prochaines insertions et modifications. Dans certains cas, cela aide MySQL si les colonnes sont dans l'ordre dans lequel vous allez trier les valeurs. Cette option n'est vraiment utile que si vous savez á l'avance dans quel ordre vous effectuerez les tris : vous y gagnerez alors en performances.
- Si vous utilisez `ALTER TABLE` sur une table `MyISAM`, tous les index non-uniques sont crèès par des opèrations sèparèes. (comme dans `REPAIR`). Cela devrait rendre `ALTER TABLE` plus rapide quand vous avez beaucoup d'index.
- Depuis la version **4.0** de MySQL, la fonctionnalitè ci-dessus peut être activèe explicitement. `ALTER TABLE ... DISABLE KEYS` force MySQL á ne plus mettre á jour les index non-uniques pour les tables au format `MyISAM`. `ALTER TABLE ... ENABLE KEYS` doit alors être utilisè pour recrèer les index manquants. Comme MySQL le fait avec un algorithme spècial qui est plus rapide que le fait d'insèrer les clefs une par une, dèsactiver les clefs peut vous faire gagner en performances.
- Avec la fonction `mysql_info()` de l'API C, vous pouvez savoir combien d'enregistrements ont ètè copiès, et (quand `IGNORE` est spècifiè) combien d'enregistrements ont ètè èffacès á cause de la clef unique.
- Les clauses `FOREIGN KEY`, `CHECK`, et `REFERENCES` ne font rien pour le moment, á part pour les tables InnoDB qui supportent la commande `ADD CONSTRAINT FOREIGN KEY (...) REFERENCES ... (...)`. Notez que InnoDB ne permet pas la spècification de `nom_index`. Voir Section 7.5 [InnoDB], page 544. La syntaxe pour les autres types de tables est fournie pour assurer la compatibilitè, rendre le port du code á partir d'autres serveurs SQL plus facile et faire fonctionner les applications qui crèent des tables avec des rèfèrences. Voir Section 1.7.4 [Differences from ANSI], page 36.

Voilá un exemple qui montre quelques utilisations de `ALTER TABLE`. On commence par une table `t1` crèèe comme suit :

```
mysql> CREATE TABLE t1 (a INTEGER,b CHAR(10));
```

Pour renommer la table de `t1` á `t2` :

```
mysql> ALTER TABLE t1 RENAME t2;
```

Pour changer une colonne `a` de `INTEGER` en `TINYINT NOT NULL` (en laissant le même nom), et pour changer une colonne `b` de `CHAR(10)` á `CHAR(20)` et la renommant de `b` en `c` :

```
mysql> ALTER TABLE t2 MODIFY a TINYINT NOT NULL, CHANGE b c CHAR(20);
```

Pour ajouter une nouvelle colonne `TIMESTAMP` nommèe `d` :

```
mysql> ALTER TABLE t2 ADD d TIMESTAMP;
```

Pour ajouter un index sur une colonne `d`, et rendre la colonne `a` la clef primaire :

```
mysql> ALTER TABLE t2 ADD INDEX (d), ADD PRIMARY KEY (a);
```

Pour effacer la colonne `c` :

```
mysql> ALTER TABLE t2 DROP COLUMN c;
```

Pour ajouter une nouvelle colonne `AUTO_INCREMENT` nommèe `c` :

```
mysql> ALTER TABLE t2 ADD c INT UNSIGNED NOT NULL AUTO_INCREMENT,
           ADD INDEX (c);
```

Notez que nous avons indexè `c`, car les colonnes `AUTO_INCREMENT` doivent être indexèes, et que nous dèfinissons aussi `c` en tant que `NOT NULL`, car les colonnes indexèes ne peuvent être `NULL`.

Quand vous ajoutez une colonne `AUTO_INCREMENT`, les valeurs de la colonne sont remplies automatiquement pour vous. Vous pouvez choisir la valeur de dèpart pour l'indexation en utilisant `SET INSERT_ID=#` avant `ALTER TABLE` ou en utilisant l'option `AUTO_INCREMENT = #` de la table. Voir Section 5.5.6 [SET OPTION], page 396.

Avec les tables de type MyISAM, si vous ne changez pas la colonne `AUTO_INCREMENT`, l'indice d'auto-incrèmentation ne sera pas affectè. Si vous effacez une colonne `AUTO_INCREMENT` puis en ajoutez une autre, l'indexation recommencera á partir de 1.

Voir Section A.6.1 [ALTER TABLE problems], page 709.

6.5.5 Syntaxe de `RENAME TABLE`

```
RENAME TABLE nom_de_table TO nouveau_nom_de_table[, nom_de_table2 TO nouveau_nom_de
```

Le changement de nom se fait atomiquement ce qui signifie qu'aucun autre processus ne peut accèder la table tant que l'opèration est en cours. Cela rend possible de remplacer une vielle table avec une table vide :

```
CREATE TABLE nouvelle_table (...);
RENAME TABLE ancienne_table TO backup_table, nouvelle_table TO ancienne_table;
```

L'opèration s'effectue de gauche á droite ce qui signifie que si vous voulez èchanger deux noms de tables, vous devez :

```
RENAME TABLE ancienne_table    TO backup_table,
             nouvelle_table    TO ancienne_table,
             backup_table TO nouvelle_table;
```

Si les deux bases de donnèes sont sur le même disque, vous pouvez renommer á travers les bases :

```
RENAME TABLE bdd_courante.nom_de_table TO autre_bdd.nom_de_table;
```

Quand vous exècutez `RENAME`, vous ne pouvez avoir aucune transaction active ou une table protègèe en mode ècriture. Vous devez avoir les priviléges `ALTER` et `DROP` sur l'ancienne table, et les priviléges `CREATE` et `INSERT` sur la nouvelle.

Si MySQL rencontre des erreurs dans un renommage multiple, il remettra les noms changès á leurs valeurs d'origine pour revenir á l'ètat d'origine.

`RENAME TABLE` a ètè ajoutè á la version 3.23.23 de MySQL.

6.5.6 Syntaxe de `DROP TABLE`

```
DROP [TEMPORARY] TABLE [IF EXISTS] nom_de_table [, nom_de_table2,...] [RESTRICT | C
```

`DROP TABLE` supprime une ou plusieurs tables. Toutes les donnèes et la structure de la tables sont *perdues*, alors soyez **prudents** avec cette commande !

Depuis la version 3.22 de MySQL, vous pouvez utiliser le mot rèservè `IF EXISTS` pour èviter l'affichage des erreurs pour les tables qui n'existent pas.

`RESTRICT` et `CASCADE` sont autorisès pour faciliter le port. Pour le moment, elles ne font rien.

Note : `DROP TABLE` validera automatiquement la transaction actives.

`TEMPORARY` est pour le moment ignorè; Dans un futur proche, il servira á s'assurer qu'on efface vraiment une table temporaire.

6.5.7 Syntaxe de `CREATE INDEX`

```
CREATE [UNIQUE|FULLTEXT] INDEX index_name
       ON tbl_name (col_name[(length)],... )
```

La requête `CREATE INDEX` n'effectue aucune action sur les versions de MySQL antèrieures á la version 3.22. Dans les versions 3.22 et supèrieures, `CREATE INDEX` est èquivalent á une requête `ALTER TABLE` pour crèer des index. Voir Section 6.5.4 [`ALTER TABLE`], page 512.

Normalement, tous les index sont crèès en même temps que la table elle même avec `CREATE TABLE`. Voir Section 6.5.3 [`CREATE TABLE`], page 504. `CREATE INDEX` permet d'ajouter des index á une table existante.

Une liste de colonnes de la forme `(col1,col2,...)` crèe un index multi-colonnes. Les valeurs de l'index sont crèèes en concatènant la valeur dex colonnes donnèes.

Pour les colonnes `CHAR` et `VARCHAR`, les index peut être crèès sur uniquement une partie de la colonne, avec la syntax `col_name(length)`.(Pour les colonnes `BLOB` et `TEXT` la longueur d'index est obligatoire.) La requête suivante crèe un index en utilisant les 10 premiers caractéres de la colonne `name`:

```
mysql> CREATE INDEX part_of_name ON customer (name(10));
```

Comme la plupart des noms ont en gènèral des diffèrences dans les 10 premiers caractéres, l'index ne devrait pas être plus lent qu'un index crèè á partir de la colonne `name` en entier. Ainsi, en n'utilisant qu'une partie de la colonne pour les index, on peut rèduire la taille du fichier d'index, ce qui peut permettre d'èconomiser beaucoup d'espace disque, et peut aussi accèlèrer les opèrations `INSERT`!

Il est important de savoir qu'on peut indexer une colonne qui peut avoir la valeur `NULL` ou une colonne `BLOB/TEXT` que si on utilise une version 3.23.2 ou supèrieure de MySQL et en utilisant le type `MyISAM`.

Pour plus d'informations á propos de l'utilisation des index dans MySQL, voir Section 5.4.3 [MySQL indexes], page 384.

Les index `FULLTEXT` ne peuvent indexer que des colonnes `VARCHAR` ou `TEXT`, et seulement dans les tables `MyISAM`. Les index `FULLTEXT` sont disponibles dans les versions 3.23.23 et supèrieures de MySQL. Section 6.8 [Fulltext Search], page 522.

6.5.8 Syntaxe de `DROP INDEX`

```
DROP INDEX nom_de_l_index ON nom_de_table
```

`DROP INDEX` supprime l'index nommè `nom_de_l_index` de la table `nom_de_table`. `DROP INDEX` ne fait rien avec la version 3.22 et les prècèdentes. Depuis cette version, `DROP INDEX` est un alias d'`ALTER TABLE` supprimant l'index.

Voir Section 6.5.4 [`ALTER TABLE`], page 512.

6.6 Commandes de bases de l'utilisateur de MySQL

6.6.1 Syntaxe de USE

```
USE db_name
```

La commande `USE db_name` spècifie á MySQL d'utiliser la base `db_name` comme base par dèfaut pour les requêtes ne les mentionnant pas. La base choisie reste la même jusqu'á la fermeture de la session ou un nouvel appel á `USE` :

```
mysql> USE db1;
mysql> SELECT COUNT(*) FROM ma_table;      # sèlectionne á partir de db1.ma_table
mysql> USE db2;
mysql> SELECT COUNT(*) FROM ma_table;      # sèlectionne á partir de db2.ma_table
```

Rendre une base de donnèes la base courante (en utilisant `USE`) ne vous interdit pas l'accés á d'autres tables dans d'autres bases. L'exemple suivant accéde á la table `author` de la base `db1` et á la table `editor` de la base `db2` :

```
mysql> USE db1;
mysql> SELECT author_name,editor_name FROM author,db2.editor
    ->        WHERE author.editor_id = db2.editor.editor_id;
```

La commande `USE` est fournie pour assurer la compatibilitè Sybase.

6.6.2 Syntaxe de DESCRIBE (obtenir des informations sur les colonnes)

```
{DESCRIBE | DESC} nom_de_table [nom_de_colonne | wild]
```

`DESCRIBE` est un raccourçi de `SHOW COLUMNS FROM`. Voir Section 4.5.6.1 [SHOW DATABASE INFO], page 268.

`DESCRIBE` fournit des informations á propos des colonnes de la table. `nom_de_colonne` peut être le nom d'une colonne ou une chaîne contenant les caractéres spèciaux SQL '`%`' et '`_`'.

Si le type de colonne est diffèrent de celui que vous pensiez avoir dèfinit lors du `CREATE TABLE`, nottez que MySQL change le type des colonnes de temps en temps. Voir Section 6.5.3.1 [Silent column changes], page 511.

Cette instruction est fournie pour une meilleure compatibilitè avec Oracle.

L'instruction `SHOW` renvoit les mêmes informations. Voir Section 4.5.6 [SHOW], page 267.

6.7 Commandes relatives aux verrous et aux transactions

6.7.1 Syntaxe de BEGIN/COMMIT/ROLLBACK

Par dèfaut, MySQL est lancè en mode `autocommit`. Cela signifie que chaque modification effectuè est enregistrè immèdiatement sur le disque par MySQL.

Si vous utilisez des tables supportant les transactions (comme `InnoDB`, `BDB`), vous pouvez configurer MySQL en mode non-`autocommit` grâce á la commande:

```
SET AUTOCOMMIT=0
```

A partir de lá, vous devez utiliser `COMMIT` pour enregistrer les modifications sur le disque ou `ROLLBACK` pour ignorer les modifications apportèes depuis le dèbut de la transaction.

Si vous souhaitez sortir du mode `AUTOCOMMIT` pour une sèrie d'opèrations, vous pouvez utiliser les commandes `BEGIN` ou `BEGIN WORK` :

```
BEGIN;
SELECT @A:=SUM(salary) FROM table1 WHERE type=1;
UPDATE table2 SET summmary=@A WHERE type=1;
COMMIT;
```

Vous devez savoir que si vous utilisez des tables ne supportant pas les transactions, les modifications seront ècrites immèdiatement, indèpendamment de l'ètat du mode de `autocommit`

Si vous faites un `ROLLBACK` aprés avoir modifiè une table non transactionnelle, vous obtiendrez (`ER_WARNING_NOT_COMPLETE_ROLLBACK`) comme message d'alerte. Toutes les tables supportant les transactions seront restaurèes, mais aucune des autres tables ne changera.

Si vous utilisez `BEGIN` ou `SET AUTOCOMMIT=0`, il est recommandè d'utiliser les "binary log" de MySQL á la place des anciens logs d'update pour les sauvegardes. Les transactions sont stockèes dans les logs binaires en un seul bloc, aprés `COMMIT`, pour être sû que les transactions qui ont ètè annulèes ne soient pas enregistrèes. Voir Section 4.9.4 [Binary log], page 329.

Les commandes suivantes dèclenchent la fin des transactions (comme si vous aviez utilisè `COMMIT` avant d'exècuter la commande):

Commande	**Commande**	**Commande**
`ALTER TABLE`	`BEGIN`	`CREATE INDEX`
`DROP DATABASE`	`DROP TABLE`	`RENAME TABLE`
`TRUNCATE`		

Vous pouvez changer le niveau d'isolation des transactions avec `SET TRANSACTION ISOLATION LEVEL ...`. Voir Section 6.7.3 [SET TRANSACTION], page 521.

6.7.2 Syntaxe de `LOCK TABLES/UNLOCK TABLES`

```
LOCK TABLES nom_de_table [AS alias] {READ [LOCAL] | [LOW_PRIORITY] WRITE}
            [, nom_de_table [AS alias] {READ [LOCAL] | [LOW_PRIORITY] WRITE} ...]
...
UNLOCK TABLES
```

`LOCK TABLES` verrouille une table pour le thread courant. `UNLOCK TABLES` dèverrouillera automatiquement tous les verrous posès par le thread courant. Toutes les tables verrouillèes par le thread courant sont automatiquement dèverrouillèes quand ce thread utilise á nouveau `LOCK TABLES`, ou quand la connexion au serveur est perdue.

L'utilisation de `LOCK TABLES` dans MySQL 4.0.2 nècessite le privilége `LOCK TABLES` global et un privilége de `SELECT` sur les tables impliquèes. Dans MySQL 3.23, il faut les priviléges `SELECT`, `insert`, `DELETE` et `UPDATE` sur les tables.

Les principales raisons d'utiliser `LOCK TABLES` sont l'èmulation de transactions ou l'accèlèration des processus de modification de tables. Cela sera dètaillè plus loin.

Si un thread obtient un verrouillage `READ` sur une table, ce thread (et tous les autres threads) peuvent uniquement accèder a cette table en lecture. Si un thread obtient un verrouillage `WRITE` sur une table, alors seul le thread qui a posè le verrou peut lire ou ècrire sur cette table. Tous les autres threads sont bloquès.

La diffèrence entre `READ LOCAL` et `READ` est que `READ LOCAL` autorise des requêtes `INSERT` non-conflictuelles á être exècutèes alors que le verrou est posè. Ceci ne peut cependant pas être utilisè si vous souhaitez modifier les fichiers de la base de donnèes en dehors de MySQL pendant que le verrou est posè.

Quand vous utilisez `LOCK TABLES`, vous devez verrouiller toutes les tables que vous allez utiliser, et vous devez utiliser les mêmes alias sur ce que vous utiliserez dans vos requêtes ! Si vous utilisez une table a plusieurs reprises dans une requête (avec des alias), vous devez verrouiller chacun des alias !

Les verrous `WRITE` ont normalement des prioritès supèrieures aux verrous `READ`, afin de s'assurer que les updates sont exècutès au plus vite. Cela signifie que si un thread demande un verrou `READ` et qu'un autre thread demande un verrou `WRITE`, la demande de verrou `READ` attendra que le thread `WRITE` ait abouti pour libèrer le verrou. Vous pouvez utiliser le verrou `LOW_PRIORITY WRITE` pour permettre á d'autres threads d'obtenir des verrous `READ` pendant que le thread attend le verrou `WRITE`. Vous ne devriez utiliser les verrous `LOW_PRIORITY WRITE` que si vous êtes sûr qu'il y aura effectivement un moment oú aucun thread ne posera de verrou `READ`.

`LOCK TABLES` fonctionne de la maniére suivante :

1. Trie toutes les tables á verrouiller dans un ordre dèfini par MySQL (l'utilisateur ne dèfinit pas d'ordre).
2. Si une table est verrouillèe avec un verrou read et un verrou write, il pose le verrou write avant le read.
3. Verrouille une table á la fois jusqu'á ce que le thread ait tous ses verrous.

Cette politique garantit le bon verrouillage des tables. Il faut cependant connaitre certaines choses sur ce schema :

Si vous utilisez un verrou `LOW_PRIORITY WRITE` pour une table, cela signifie seulement que MySQL attendra, pour poser ce verrou, qu'aucun autre thread ne rèclame de verrou `READ`. Quand le thread aura le verrou `WRITE` et qu'il attendra que les verrous soient posès sur les autres tables de la liste, tous les autres threads attendront que le verrou `WRITE` soit libèrè. Si cela devient un probléme grave pour votre application, il est conseillè de convertir des tables en tables supportant les transactions.

Vous pouvez terminer un thread attendant un verrouillage de table en toute sècuritè avec `KILL`. Voir Section 4.5.5 [KILL], page 266.

Il est **dèconseillè** de verrouiller des tables utilisèes avec `INSERT DELAYED`, car, dans ce cas, la requête `INSERT` est exècutèe dans un autre thread.

Normalement, vous n'avez pas besoin de verrouiller les tables puisque chaque requête `UPDATE` est atomique : aucun autre thread ne peut interfèrer avec une autre requête active. Il existe cependant quelques cas oú vous aurez besoin de verrouiller les tables :

- Si vous allez exècuter plusieurs requêtes sur plusieurs tables, il est prèfèrable, d'un point de vue rapiditè, de verrouiller les tables dont vous aurez besoin. L'inconvènient,

bien sur, est que les autres threads ne pourront pas intervenir sur ces tables durant vos opèrations, ni en extraire des informations si la table est en `WRITE`-locked.

La raison pour laquelle les requêtes sont plus rapides avec `LOCK TABLES` est que MySQL ne rafraichit pas l'index des clès des tables verrouillèes tant que `UNLOCK TABLES` n'est pas invoquè (normalement, le cache des clès est rafraichi aprés chaque requête SQL). Cela accelére les insertions, les modifications et les suppressions de donnèes dans les tables `MyISAM`.

- Si vous utilisez un type de table dans MySQL qui ne supporte pas les transactions, vous devez utiliser `LOCK TABLES` pour vous assurez qu'aucun autre thread ne s'intercale entre un `SELECT` et un `UPDATE`. L'exemple suivant necessite `LOCK TABLES` pour s'executer en toute sècuritè :

```
mysql> LOCK TABLES trans READ, customer WRITE;
mysql> SELECT SUM(value) FROM trans WHERE customer_id=some_id;
mysql> UPDATE customer SET total_value=sum_from_previous_statement
    ->        WHERE customer_id=some_id;
mysql> UNLOCK TABLES;
```

Sans `LOCK TABLES`, Il est possible qu'un autre thread ait inserè une nouvelle ligne dans la table `trans` entre l'execution du `SELECT` et l'exècution de la requête `UPDATE`.

L'utilisation de modifications incrèmentales (`UPDATE customer SET value=value+nouvelle_valeur`) ou de la fonction `LAST_INSERT_ID()` permet de se passer de `LOCK TABLES` en de nombreuses occasions.

Il est aussi possible de rèsoudre de nombreux cas en utilisant un verrou utilisateur, avec les fonctions `GET_LOCK()` et `RELEASE_LOCK()`. Ces verrous sont stockès dans une table de hashage dans le serveur et utilisent les fonctions `pthread_mutex_lock()` et `pthread_mutex_unlock()` pour plus de vitesse. Voir Section 6.3.6.2 [Miscellaneous functions], page 472.

Voir Section 5.3.1 [Internal locking], page 380 pour plus de dètails.

Il est possible de verrouiller tous les tables de toutes les bases avec la commande `FLUSH TABLES WITH READ LOCK`.

Voir Section 4.5.3 [FLUSH], page 265. C'est une mèthode trés pratique pour effectuer des sauvegardes si vous utilisez un systéme de fichiers qui, comme Veritas, permet de crèer des instantanès.

NOTE: `LOCK TABLES` ne fonctionne pas avec les transactions et validera automatiquement toutes les transactions actives avant de poser verrouiller la table.

6.7.3 Syntaxe de `SET TRANSACTION`

```
SET [GLOBAL | SESSION] TRANSACTION ISOLATION LEVEL
{ READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ | SERIALIZABLE }
```

Configuration du niveau d'isolation des transactions en gènèral, pour la totalitè de la session, ou pour la prochaine transaction.

Le comportement par dèfaut est de configurer le niveau d'isolation pour la transaction suivante (qui n'a pas encore ètè commencèe) En utilisant le paramétre `GLOBAL`, on configure le niveau par dèfaut global pour toutes les nouvelles connections. Cette commande requiert

les priviléges SUPER. En utilisant le paramétre SESSION, on configure le niveau par dèfaut pour toutes les prochaines transactions effectuèes durant la session actuelle.

On peut configurer le niveau d'isolation global des transactions pour mysqld avec --transaction-isolation=.... Voir Section 4.1.1 [Command-line options], page 192.

6.8 Recherche en Texte-entier (Full-text) dans MySQL

Comme dans la version 3.23.23, MySQL propose l'indexation et la recherche sur l'ensemble d'un champ "text" (full-text). Les index sur le texte en entier de MySQL sont des index de type FULLTEXT. Les index FULLTEXT sont utilisès avec les tables MyISAM et peuvent être crèès depuis des colonnes de types CHAR, VARCHAR, ou TEXT au moment de CREATE TABLE ou plus tard avec ALTER TABLE ou CREATE INDEX. Pour les enregistrements les plus grands, il sera plus rapide de charger les donnès dans une table qui n'a pas d'index FULLTEXT, et ensuite de crèer l'index avec ALTER TABLE (ou CREATE INDEX). L'enregistrement de données dans une table qui a dèjá des index FULLTEXT sera plus lent.

La recherche sur un texte en entier est effectuèe par la fonction MATCH().

```
mysql> CREATE TABLE articles (
    ->   id INT UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
    ->   title VARCHAR(200),
    ->   body TEXT,
    ->   FULLTEXT (title,body)
    -> );
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO articles VALUES
    -> (NULL,'MySQL Tutorial', 'DBMS stands for DataBase ...'),
    -> (NULL,'How To Use MySQL Efficiently', 'After you went through a ...'),
    -> (NULL,'Optimising MySQL','In this tutorial we will show ...'),
    -> (NULL,'1001 MySQL Tricks','1. Never run mysqld as root. 2. ...'),
    -> (NULL,'MySQL vs. YourSQL', 'In the following database comparison ...'),
    -> (NULL,'MySQL Security', 'When configured properly, MySQL ...');
Query OK, 6 rows affected (0.00 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM articles
    ->          WHERE MATCH (title,body) AGAINST ('database');
+----+-------------------+------------------------------------------+
| id | title             | body                                     |
+----+-------------------+------------------------------------------+
|  5 | MySQL vs. YourSQL | In the following database comparison ... |
|  1 | MySQL Tutorial    | DBMS stands for DataBase ...             |
+----+-------------------+------------------------------------------+
2 rows in set (0.00 sec)
```

La fonction MATCH() effectue la recherche d'une chaîne de caractéres dans une liste de textes (et dans un groupe d'une ou plusieurs colonnes utilisèes pour l'index FULLTEXT). La

chaîne recherchèe est donnèe en argument á `AGAINST()`. La recherche est sans distinguer les majuscules des minuscules. Pour chaque ligne de la table, `MATCH()` retourne une valeur de pertinence, qui est une mesure de la ressemblance entre le chaîne recherchèe et le texte de la ligne dans le colonne donnèe dans la liste de `MATCH()`.

Quand `MATCH()` est utilisè comme condition de `WHERE` (voir l'exemple suivant) les lignes retournèes sont automatiquement organisèes avec la pertinence la plus èlevè en premier. Ces pertinences sont des nombres non nègatifs en virgule flottante. Une pertinence de zèro signifie qu'il n'y a pas de similaritè. La pertinence est calculè en fonction du nombre de mots dans la ligne, du nombre de mots uniques dans cette ligne, du nombre total de mots dans la liste, et du nombre de documents (lignes) qui contiennent un mot en particulier.

Il est aussi possible d'exècuter une recherche en mode boolèen. Ceci est dècrit dans une section ultèrieure.

L'exemple prècèdent est une illustration èlèmentaire qui montre comment on utilise la fonction `MATCH()`. Les lignes sont retournèes par ordre dècroissant de pertinence.

L'exemple suivant montre comment rècupèrer la valeur de pertinence explicitement. Comme il n'y a pas de condition `WHERE` ni de condition `ORDER BY` les lignes retournèes ne sont pas ordonnèes.

```
mysql> SELECT id,MATCH (title,body) AGAINST ('Tutorial') FROM articles;
+----+----------------------------------------+
| id | MATCH (title,body) AGAINST ('Tutorial') |
+----+----------------------------------------+
|  1 |                        0.64840710366884 |
|  2 |                                       0 |
|  3 |                        0.66266459031789 |
|  4 |                                       0 |
|  5 |                                       0 |
|  6 |                                       0 |
+----+----------------------------------------+
6 rows in set (0.00 sec)
```

L'exemple suivant est plus complexe. La requête retourne la valeur de pertinence et organise les lignes par ordre dècroissant de pertinence. Pour obtenir ce rèsultat, il faut spècifier `MATCH()` deux fois. Cela ne cause pas de surcharge car l'optimisateur de MySQL remarquera que les deux appels á `MATCH()` sont identiques et appellent le code de recherche sur texte plein une seule fois.

```
mysql> SELECT id, body, MATCH (title,body) AGAINST
    -> ('Security implications of running MySQL as root') AS score
    -> FROM articles WHERE MATCH (title,body) AGAINST
    -> ('Security implications of running MySQL as root');
+----+-------------------------------------+-----------------+
| id | body                                | score           |
+----+-------------------------------------+-----------------+
|  4 | 1. Never run mysqld as root. 2. ... | 1.5055546709332 |
|  6 | When configured properly, MySQL ... |   1.31140957288 |
+----+-------------------------------------+-----------------+
2 rows in set (0.00 sec)
```

MySQL utilise un filtre trés simple pour sèparer le texte en mots. Un "mot" est n'importe quelle chaîne de caractéres constituèe de lettres, chiffres, ‘’’ et ‘_’. Tout "mot" prèsent dans la liste des mots á ignorès ou qui est trop court (3 caractéres ou moins) est ignorè.

Tous les mots corrects dans la liste et dans la requête sont pondèrès en fonction de leur importance dans la liste ou la requête. de cette façon, un mot prèsent dans de nombreux documents aura un poids faible (et peut être même un poids nul), car il a peut d'importance dans cette requête particuliéte. Au contraire, si le mot est rare, il recevra un poids fort. Le poids des mots sont alors rassemblès pour calculer la pertinence de la ligne.

Une telle technique fonctionne plus efficacement sur de grands volumes de donnèes (en fait, elle est optimisèe pour cela). Avec des toutes petites tables, la distribution des mots ne refléte par correctement leur valeur sèmantique et ce modéle peut parfois produire des rèsultats ètranges.

```
mysql> SELECT * FROM articles WHERE MATCH (title,body) AGAINST ('MySQL');
Empty set (0.00 sec)
```

La recherche du mot `MySQL` ne donne aucun rèsultat dans l'exemple prècèdent, car il est prèsent dans plus de la moitiè des lignes. Ainsi, il est considèrè comme un mot á ignorer (un mot avec une valeur sèmantique nulle). C'est le comportement le plus optimal – un langage de requêtes ne doit pas retourner chaque ligne d'une table de 1 GB.

Un mot qui est trouvè dans la moitiè des enregistrements d'une table n'est pas efficace pour trouver les document appropriès. En fait, il trouvera sûrement beaucoup de documents inappropriès á la recherche. On sait tous que cela arrive souvent lorsqu'on recherche quelque chose sur internet en utilisant un moteur de recherche. C'est en suivant ce raisonnement que ces lignes se sont vues attribuer une valeur sèmantique trés basse dans **ce cas particulier**.

Avec la version 4.0.1, MySQL peut aussi performer des recherches boolèennes en utilisant le modificateur `IN BOOLEAN MODE`.

```
mysql> SELECT * FROM articles WHERE MATCH (title,body)
    ->     AGAINST ('+MySQL -YourSQL' IN BOOLEAN MODE);
+----+-------------------------------+---------------------------------------+
| id | title                         | body                                  |
+----+-------------------------------+---------------------------------------+
|  1 | MySQL Tutorial                | DBMS stands for DataBase ...          |
|  2 | How To Use MySQL Efficiently  | After you went through a ...          |
|  3 | Optimising MySQL              | In this tutorial we will show ...     |
|  4 | 1001 MySQL Tricks             | 1. Never run mysqld as root. 2. ...   |
|  6 | MySQL Security                | When configured properly, MySQL ...   |
+----+-------------------------------+---------------------------------------+
```

Cette requête a retournè toutes les lignes qui contiennent le mot `MySQL` (note : la barriére des 50% n'est pas utilisèe), mais ne contiennent **pas** le mot `YourSQL`. Notez qu'une recherche en mode boolèen ne trie pas automatiquement les lignes par ordre de pertinence. Vous pouvez le voir dans le rèsultat de la requête prècèdente, oú la ligne avec la plus haute pertinence (celui qui contient `MySQL` deux fois) est listè en dernier. Une recherche boolèenne peu aussi fonctionner sans un index `FULLTEXT`, même si cela risque de **ralentir** le processus.

La recherche boolèenne supporte les opèrateurs suivants :

`+` Un signe plus avant un mot indique que celui-ci **doit** être prèsent dans chaque ligne retournèe.

`-` Un signe moins avant un mot indique que celui-ci **ne doit pas** être prèsent dans chaque ligne retournèe.

Par dèfaut (quand ni moins ni plus n'est spècifiè) le mot est optionnel, mais la ligne qui le contient sera mise en valeur. Ce fonctionnement est celui de `MATCH() ... AGAINST()` dans le paramétre `IN BOOLEAN MODE`.

`< >` Ces deux opèrateurs servent á faire varier la contribution d'un mot á la pertinence associèe á une ligne. L'opèrateur `<` diminue la contribution alors que `>` l'augmente. Voir l'exemple ci-dessous.

`( )` Les parenthéses sont utilisèes pour grouper des mots dans une sous-expression.

`~` Un tilde avant un mot opére comme un opèrateur de nègation, ce qui rend la contribution du mot á la pertinence nègative. C'est utile pour se dèbarrasser des mots inutiles. Une ligne contenant un mot comme cela sera moins mise en avant que les autres, mais ne sera pas exclue, ce qui serait le cas avec l'opèrateur `-` .

`*` C'est l'opèrateur de tronquage. Contrairement aux autres opèrateurs, il doit **suivre** le mot, non le prècèder.

`"` La phrase, protègèe par des guillemets `"`, ne trouvera que les lignes contenant cette phrase **littèralement, comme elle a ètè entrèe**.

et voici quelques exemples :

`apple banana`
trouve les lignes qui contiennent au moins l'un de ces mots.

`+apple +juice`
... les deux mots.

`+apple macintosh`
... le mot "apple", mais elles sont plus valorisèes si elles contiennent aussi "macintosh".

`+apple -macintosh`
... le mot "apple" mais pas le mot "macintosh".

`+apple +(>pie <strudel)`
... "apple" et "pie", ou "apple" et "strudel" (l'ordre n'importe pas), mais valorise plus "apple pie" que "apple strudel".

`apple*` ... "apple", "apples", "applesauce", et "applet".

`"some words"`
... "some words of wisdom", mais pas "some noise words".

6.8.1 Restrictions avec full-text

- Tous les paramétres de la fonction `MATCH()` doivent être des colonnes de la même table faisant partie du même index `FULLTEXT` index, sauf si `MATCH()` est en mode `BOOLEAN`.

- Les arguments de `MATCH()` doivent correspondre exactement á la liste de colonnes de certaines dèfinitions d'index `FULLTEXT` pour la table, sauf si `MATCH()` est utilisè dans un contexte `BOOLEAN`.
- L'argument de `AGAINST()` doit être une chaîne constante.

6.8.2 Paramètrage prècis de la recherche Full-text de MySQL

La recherche sur texte entier n'a malheureusement pas encore beaucoup de paramétres modifiables par l'utilisateur, même si l'ajout de certains apparaît trés haut dans le TODO. Si vous utilisez MySQL depuis les sources (voir Section 2.3 [Installing source], page 84), vous pouvez mieux contrôler le fonctionnement de la recherche sur texte entier.

La recherche sur texte entier a ètè paramètrèe pour une efficacitè de recherche maximale. La modification du comportement par dèfaut ne fera gènèralement que diminuer la qualitè des rèsultats des recherches. Il ne faut pas modifier les sources de MySQL sans savoir prècisèment ce qu'on fait.

- La taille minimale des mots á indexer est dèfinie dans la variable `ft_min_word_len` de MySQL. Voir Section 4.5.6.4 [SHOW VARIABLES], page 273. Vous pouvez modifier cette valeur pour celle que vous prèfèrez, puis reconstruire les index `FULLTEXT`. (Cette variable n'existe que pour la version 4.0 de MySQL)
- La liste des mots rejetès est dèfinie dans le fichier '`myisam/ft_static.c`'. Modifiez le selon vos goûts, recompilez MySQL et reconstruisez vos index `FULLTEXT`.
- Le taux de 50% est dèterminè par la mèthode de pondèration choisie. Pour le dèsactiver, il faut changer la ligne suivante dans '`myisam/ftdefs.h`':

  ```
  #define GWS_IN_USE GWS_PROB
  ```

  Par la ligne:

  ```
  #define GWS_IN_USE GWS_FREQ
  ```

  Puis recompiler MySQL. Il n'est pas nècessaire de reconstruire les index dans ce cas. **Note**: En faisant ces modifications, vous diminuez **ènormèment** les capacitès de MySQL á fournir des valeurs pertinentes pour la fonction `MATCH()`. Si vous avez rèellement besoin de faire des recherches avec ces mots courants, il est prèfèrable de rechercher `EN MODE BOOLEEN`, lequel ne respecte pas le taux de 50%.
- Parfois le gestionnaire du moteur de recherche voudrait changer les opèrateurs utilisès pour les recherches boolèennes sur des textes entiers. Ceux ci sont dèfinis dans la variable `ft_boolean_syntax`. Voir Section 4.5.6.4 [SHOW VARIABLES], page 273. Cependant, cette variable n'est pas modifiable, sa valeur est fixèe dans '`myisam/ft_static.c`'.

Pour les modifications qui nècessitent une reconstruction des index `FULLTEXT`, la mèthode la plus simple pour les tables MyISAM est l'opèration suivante, qui reconstruit les fichiers d'index :

```
mysql> REPAIR TABLE nom_de_table USE_FRM;
```

6.8.3 A faire dans la recherche Full-text

- Rendre toutes les opèrations avec l'index `FULLTEXT` **plus rapides**.

- Opèrateurs de proximitè
- Support de listes de mots á toujours indexer ("always-index words"). Ceux ci pourraient être n'importe quelle chaîne de caractéres que l'utilisateur voudrais traiter comme des mots: par exemple "C++", "AS/400", "TCP/IP", etc.
- Support de la recherche full-text sur les tables `MERGE`.
- Support des jeux de caractéres multi-octets.
- Rendre la liste des mots ignorès dèpendante de la langue des donnèes.
- Stemming (dèpendante de la langue des donnèes, bien sûr).
- Prè-parseur gènèrique pour les UDF fournies par l'utilisateur.
- Rendre le modéle plus flexible (en ajoutant des valeurs paramètrables pour `FULLTEXT` dans `CREATE/ALTER TABLE`).

6.9 Cache de requêtes MySQL

Depuis la version 4.0.1, le `MySQL server` bènèficie d'un `cache de requêtes`. En fait, le cache sauvegrade le texte d'une requête `SELECT` avec le rèsultat qui a ètè envoyè au client. Si une requête identique est appelèe par la suite, le serveur retournera le rèsultat á partir du cache plutôt que d'analyser puis exècuter la requête á nouveau.

NOTE : Le cache de requêtes ne retourne pas de donnèes pèrimèes. A chaques fois que les donnèes sont modifièes, les entrèes correspondantes dans le cache sont effacèes.

La cache de requêtes est trés utile dans des applications oú (quelques) tables ne changent pas souvent et que vous avez les mêmes requêtes qui s'y rèpètent. C'est une solution typique pour les webmestres dont les sites utilisent du contenu dynamique.

Voici quelques performances du cache de requêtes. (Ces rèsultats on ètè gènèrès en utilisant la suite benchmark MySQL sur un Linux Alpha 2 x 500 MHz avec 2GB RAM et un cache de requêtes de 64MB) :

- Si toutes les requêtes que vous effectuez sont simples (comme sèlèctionner un champ d'une table n'en contenant qu'un) mais différent d'une maniére que toutes les requêtes ne peuvent être cachèes, le gain lors de l'utilisation du cache est de 13%. Cela peut être considèrè comme le pire des cas. En rèalitè, les requêtes sont plus compliquèes que notre exemple le gain est donc plus petit.
- Les recherches sur une colonne dans une table n'en contenant qu'une sont 238% plus rapides. Cela peut être considèrè comme le gain minimal á attendre pour une requête cachèe.
- Si vous ne voulez pas utiliser le cache de requêtes paramètrez `query_cache_size` á zèro. En dèsactivant le cache de requête, il n'y a aucune surcharge apparente. (le cache de requêtes peut être dèsactivè á l'aide de l'option de configuration `--without-query-cache`)

6.9.1 Comment fonctionne le cache de requêtes

Les requêtes sont comparèes avant d'être analysèes :

```
SELECT * FROM table
```

et

```
Select * from table
```

sont considèrèes comme des requêtes diffèrentes par le cache, ce qui fait que les requêtes doivent être les mêmes (caractére á caractére) pour être considèrèes comme identiques. De plus, les requêtes peuvent être considèrèes comme diffèrentes si par exemple, un client utilise un nouveau format de protocole de communication ou un jeu de caractéres diffèrent d'un autre client.

Les requêtes qui utilisent diffèrentes bases de donnèes, diffèrentes versions de protocole ou diffèrents jeux de caractéres par dèfaut sont considèrèes comme diffèrentes et mises en cache sèparèment.

Le cache ne fonctionne pas pour les requêtes de type `SELECT CALC_ROWS ...` et `SELECT FOUND_ROWS() ...` car le nombre de lignes retournèes est aussi mis en cache.

Si une table change (`INSERT`, `UPDATE`, `DELETE`, `TRUNCATE`, `ALTER` ou `DROP TABLE|DATABASE`), alors toutes les requêtes mises en cache qui utilisaient cette table deviennent obsolétes et en sont retirèes.

Les tables transactionnelles `InnoDB` qui ont ètè modifièes seront rendues obsolétes lorsqu'un `COMMIT` sera exècutè.

Une requête ne peut être mise en cache si elle contient l'une des fonctions suivantes :

Fonction	**Fonction**	**Fonction**
Fonctions dèfinies par l'utilisateur	CONNECTION_ID	FOUND_ROWS
GET_LOCK	RELEASE_LOCK	LOAD_FILE
MASTER_POS_WAIT	NOW	SYSDATE
CURRENT_TIMESTAMP	CURDATE	CURRENT_DATE
CURTIME	CURRENT_TIME	DATABASE
ENCRYPT (avec un seul paramétre)	LAST_INSERT_ID	RAND
UNIX_TIMESTAMP (sans paramétres)	USER	BENCHMARK

Une requête ne peut être mise en cache non plus, si elle utilise une variable dèfinie par l'utilisateur, si elle opére sur les tables systéme de MySQL, est de la forme `SELECT ... IN SHARE MODE` ou de la forme `SELECT * FROM AUTOINCREMENT_FIELD IS NULL` (pour obtenir l'index du champ `auto-increment` - utilisè par ODBC).

Par contre, `FOUND ROWS()` retournera une valeur correcte, même si la requête prècèdente á utilisè le cache.

Dans les cas oú la requête n'utilise aucune table, en utilise une temporaire, ou que l'utilisateur á un droit particulier sur l'une des tables concernèes, celle ci ne sera pas mise en cache.

Avant chaque lecture á partir du cache, MySQL vèrifie que l'utilisateur á le droit de lecture (SELECT) sur toutes les bases et les tables concernèes. Si ce n'est pas le cas, le cache ne sera pas utilisè.

6.9.2 Configuration du cache de requêtes

Le cache de requête ajoute quelques variables systéme `MySQL` liées á `mysqld` qui peuvent être spècifièes dans un fichier de configuration, en ligne de commande lors du dèmarrage de `mysqld`.

- `query_cache_limit` Ne pas cacher les rèsultats dont la taille est supèrieure á cette valeur. (1 Mo par dèfaut).
- `query_cache_size` La mèmoire allouèe pour la mise en cache des requêtes. Si ce paramétre est á 0, le cache de requêtes est dèsactivè (valeur par dèfaut).
- `query_cache_type` Peut être (valeurs numèriques seulement) :

Option	Description
0	(OFF, ne pas utiliser le cache)
1	(ON, mettre en cache tous les rèsultats á part les requêtes `SELECT SQL_NO_CACHE ...`)
2	(DEMAND, mettre en cache seulement les rèsultats des requêtes `SELECT SQL_CACHE ...`)

La configuratoin du cache de requête peut être changèe durant la connexion. La syntaxe est la suivante :

```
QUERY_CACHE_TYPE = OFF | ON | DEMAND QUERY_CACHE_TYPE = 0 | 1 | 2
```

Option	Description
0 ou OFF	Ne pas utiliser le cache.
1 ou ON	mettre en cache tous les rèsultats á part les requêtes `SELECT SQL_NO_CACHE ...`.
2 ou DEMAND	mettre en cache seulement les rèsultat des requêtes `SELECT SQL_CACHE ...`.

6.9.3 Options relatives au cache de requêtes dans un SELECT

Il y'a deux options relatives au cache de requêtes qui peuvent être utilisèes dans une requête `SELECT` :

Option	Description
`SQL_CACHE`	Si `QUERY_CACHE_TYPE` est `DEMAND`, permet á la requête d'être mise en cache. Si `QUERY_CACHE_TYPE` est `ON`, ceci est la valeur par dèfaut. Si `QUERY_CACHE_TYPE` est `OFF`, ne fait rien.
`SQL_NO_CACHE`	Rend cette requête non-cachable, ne permet pas á cette requête d'être mise en cache.

6.9.4 Status du cache de requêtes et maintenance

Avec la commande `FLUSH QUERY CACHE` vous pouvez dèfragmenter le cache de requêtes pour mieux en utiliser la mèmoire. Cette commande n'effacera aucune requête du cache. `FLUSH TABLES` dèfragmente aussi le cache de requêtes.

La commande `RESET QUERY CACHE` efface tous les rèsultats de requêtes du cache.

Vous pouvez visualiser les performances du cache de requêtes avec `SHOW STATUS`:

Variable	Description

`Qcache_queries_in_cache`	Nombre de requêtes mises en cache.
`Qcache_inserts`	Nombre de requêtes ajoutèes au cache.
`Qcache_hits`	Nombre d'appel au cache.
`Qcache_lowmem_prunes`	Nombre de requêtes ôtèes du cache pour cause de mèmoire insuffisante.
`Qcache_not_cached`	Nombre de requêtes qui n'ont pas ètè mises en cache. (not cachable, or due to `QUERY_CACHE_TYPE`).
`Qcache_free_memory`	Quantitè de mèmoire libre pour le cache de requêtes.
`Qcache_total_blocks`	Nombre total de blocs dans le cache de requêtes.
`Qcache_free_blocks`	Nombre de blocs á mèmoire libre dans le cache de requêtes.

Nombre total de requêtes = `Qcache_inserts + Qcache_hits + Qcache_not_cached`.

Le cache de requêtes utilise des blocs de longueur variable, ce qui fait que `Qcache_total_blocks` et `Qcache_free_blocks` peuvent indiquer une fragmentation de la mèmoire du cache. Aprés un appel á `FLUSH QUERY CACHE` un seul (grand) bloc libre subsiste.

Note : Chaque requête a besoin au minimum de deux blocs (un pour le texte de la requête et un autre, ou plus, pour le rèsultat). De même, chaque table utilisèe par une requête a besoin d'un bloc, mais si deux ou plusieurs requêtes utilisent la même table, seul un bloc a besoin d'être allouè.

Vous pouvez aussi utiliser la variable `Qcache_lowmem_prunes` pour ajuster la taille du cache de requêtes.

7 Types de tables MySQL

Depuis la version 3.23.6 de MySQL, vous pouvez choisir entre trois formats basique de tables (`ISAM`, `HEAP` et `MyISAM`). Les nouvelles versions de MySQL peuvent supporter d'autres types de tables (`InnoDB`, ou `BDB`), cela dèpend de comment vous l'avez compilè.

Lorsque vous crèez une nouvelle table,vous pouvez dire á MySQL quel type de table il doit utiliser pour celle-ci. MySQL crèera toujours un fichier '`.frm`' pour stocker les dèfinitions de la table et des colonnes. Selon le type de table, les index et les donnèes seront stockès dans d'autres fichiers.

Notez que pour utiliser les tables `InnoDB` vous devez au moins utiliser l'option de dèmarrage `innodb_data_file_path`. Voir Section 7.5.2 [InnoDB start], page 545.

Le type de table par dèfaut de MySQL est `MyISAM`. Si vous essayez d'utiliser un type de table qui n'est pas compilèe ou activèe, MySQL crèera á la place une table de type `MyISAM`. C'est une fonctionnalitè trés utile quand vous voulez copier des tables entre diffèrents serveurs SQL qui ne supportent pas les mêmes types de tables (comme copier des tables vers un esclave qui est optimisè pour la vitesse en ne supportant pas les tables transactionnelles). Ce changement de table automatique peut toutefois induire en erreur les nouveaux utilisateurs de MySQL. Nous allons introduire des messages d'avertissement dans MySQL 4.0 et les afficher lors des transtypages automatiques des tables.

Vous pouvez changer les types de tables en utilisant la commande `ALTER TABLE`. Voir Section 6.5.4 [`ALTER TABLE`], page 512.

Notez que MySQL supporte deux diffèrents types de tables : tables transactionnelles (`InnoDB` et `BDB`) et tables non-transactionnelles (`HEAP`, `ISAM`, `MERGE`, et `MyISAM`).

Les avantages des tables transactionnelles (TST) sont :

- Plus sûr. Même si MySQL crashe ou que vous avez un probléme matèriel, vous pouvez rècupèrer vos donnèes, soit par un recouvrement automatique, soit á partir d'une sauvegarde combinèe avec le log des transactions.
- Vous pouvez combiner plusieurs commandes et les accepter toutes d'un seul coup avec la commande `COMMIT`.
- Vous pouvez utiliser `ROLLBACK` pour ignorer vos modifications (si vous n'êtes pas en mode auto-commit).
- Si une mise á jour èchoue, tout vos changements seront annulès. (Avec les tables NTST tous les changements opèrès sont permanants)

Avantages des tables non-transactionnelles (NTST) :

- Plus rapides puisqu'il n'y a pas de traitement des transactions.
- Utilisent moins d'espace disque puisqu'il n'y a pas de traitement des transavctions.
- Utilisent moins de mèmoires pour les mises á jour.

Vous pouvez combiner les tables TST et NTST dans la même requête pour obtenir le meilleur des deux types.

7.1 Tables MyISAM

MyISAM est le type par dèfaut de table en MySQL version 3.23. Il est basè sur ISAM et ajoute de nombreuses extensions pratiques.

L'index est stockè dans un fichier avec l'extension '.MYI' (MYIndex), et les donnèes sont stockèes dans un fichier avec l'extension '.MYD' (MYData). Vous pouvez vèrifier et rèparer les tables MyISAM avec myisamchk. Voir Section 4.4.6.7 [Crash recovery], page 253. Vous pouvez compresser les tables MyISAM avec myisampack, pour gagner de l'espace disque. Voir Section 4.7.4 [myisampack], page 297.

Voici les nouveautès des tables MyISAM :

- Il y a un sèmaphore dans le fichier MyISAM qui indique si la table a ètè correctement fermèe. Si mysqld est lancè avec l'option --myisam-recover, les tables MyISAM vont automatiquement être vèrifièes et rèparèes, si elles n'ont pas ètè correctement refermèes.
- Vous pouvez insèrer de nouvelles lignes dans une table qui n'a aucun bloc vide dans le fichier de donnèes, en même temps que d'autres threads lisent le fichier de donnèes (insertion simultanèe). Un bloc vide peut provenir d'une modification de ligne á format dynamique (les donnèes sont maintenant plus petites). Lorsque tous les blocs vide sont á nouveau utilisès, les insertions suivantes peuvent être simultanèes.
- Support des grands fichiers (63 bits) sur les systémes de fichiers et les systémes d'exploitation qui supportent les grands fichiers.
- Toutes les donnèes sont stockèes avec l'octet de poids faible en premier. Cela rend les donnèes indèpendantes de la machine et du systéme d'exploitation. La seule condition est que la machine utilise des entiers complètès á deux (comme toutes les machines depuis les 20 derniéres annèes), et le format IEEE pour les nombres á virgule flottante (aussi rèpandu dans les machine mainstream). Le seul domaine oú les machines ne supporteront pas la compatibilitè binaire sont les systémes embarquès (car ils ont parfois des processeurs spèciaux).

  Il n'est pas pènalisant de stocker les donnèes avec l'octet de poids faible en premier. Les octets d'une ligne de table sont gènèralement non alignès, et cela ne prend pas beaucoup de ressources de lire des octets non alignès en ordre ou en ordre inverse. Le code qui lit les donnèes dans une colonne n'est pas gourmand en temps, comparè á d'autres opèrations.
- Toutes les clès numèriques sont stockèes avec l'octet de poids fort en premier, pour amèliorer la compression.
- La gestion interne des colonnes AUTO_INCREMENT. MyISAM va automatiquement modifier cette valeur lors d'une insertion ou d'une modification. La valeur courante d'AUTO_INCREMENT peut être modifièe avec myisamchk. Cela va rendre les colonnes AUTO_INCREMENT plus rapide (au moins 10%) et les anciens nombres ne seront pas rèutilisès, comme avec les vieilles tables ISAM. Notez que lorsque une clè AUTO_INCREMENT est dèfinie á la fin d'une clè multiple, le vieux comportement est toujours prèsent.
- Lorsqu'insèrèe dans un ordre triè (comme lorsque vous utilisez une colonne de type AUTO_INCREMENT), l'arbre des clè sera scindè, pour que noeud principal ne contienne qu'une clè. Cela va amèliorer l'utilisation d'espace dans l'arbre des clès.
- BLOB et TEXT peuvent être indexès.

- Les valeurs `NULL` sont autorisèes dans une colonne indexèe. Elles prennent 0 á 1 octets par clè.
- La taille maximale d'une clè est de 500 octets par dèfaut (cela peut être modifiè en recompilant). Dans le cas des clès plus grandes que 250 octets, un bloc de clè plus grand que le bloc de 1024 octets est utilisè.
- Le nombre maximal de clè par table est par dèfaut de 32. Cela peut être agrandi á 64, sans recompiler, avec `myisamchk`.
- `myisamchk` va marquer les tables comme vèrifièe si il est exècutè avec l'option ate. `myisamchk --fast` va uniquement vèrifier les tables qui n'ont pas cette marque.
- `myisamchk -a` stocke les statistique pour les parties de clès (et non plus pour les clès complétes, comme en `ISAM`).
- Des lignes á format dynamique seront bien moins fragmentèe lorsque vous mêlez des insertions et des modifications. Cela est fait automatiquement en combinant des blocs effacès adjacent, et en ètendant ces blocs au prochain bloc vide.
- `myisampack` peut compresser des colonnes `BLOB` et `VARCHAR`.
- Vous pouvez mettre les fichiers d'index et de donnèes dans diffèrents dossiers pour gagner de la vitesse (avec l'option `DATA/INDEX DIRECTORY="path"` de `CREATE TABLE`). Voir Section 6.5.3 [CREATE TABLE], page 504.

`MyISAM` supporte aussi les fonctionnalitès suivantes, dont MySQL pourra profiter sous peu :

- Support du vrai type `VARCHAR`; une colonne `VARCHAR` commence avec une taille, stockèe sur 2 octets.
- Les tables ayant des colonnes `VARCHAR` peuvent avoir un format de lignes fixe ou dynamique.
- `VARCHAR` et `CHAR` peuvent prendre jusqu'á 64ko. Tous les segments de clè ont leur propre dèfinition de langue. Cela permettra á MySQL d'avoir des dèfinitions de langue diffèrentes pour chaque colonne.
- Un index de hachage peut être utilisè pour les index de type `UNIQUE`. Cela vous permettra d'avoir l'attribut `UNIQUE` sur n'importe quelle combinaison de colonne de votre table (vous ne pouvez pas faire de recherche sur un tel index `UNIQUE`).

Notez que les fichiers d'index sont gènèralement plus plus petit avec les tables `MyISAM` qu'avec les tables `ISAM`. Cela siginifie que `MyISAM` va noramelement utiliser moins de ressources systémes que `ISAM`, mais aura besoin de plus de processeur lors de l'insertion dans un index compressè.

Les options suivantes de `mysqld` peuvent être utilisèe pour modifier le comportement des tables `MyISAM`. Voir Section 4.5.6.4 [SHOW VARIABLES], page 273.

Option	**Description**
`--myisam-recover=#`	Rèparation automatique des tables crashèes.
`-O myisam_sort_buffer_size=#`	Buffer utilisè lors de la rèparation des tables.
`--delay-key-write=ALL`	Ne pas ècrire les buffers de clès entre deux ècriture dans une table `MyISAM`.
`-O myisam_max_extra_sort_file_size=#`	Utilisè pour aider MySQL á dècider quand utiliser la mèthode du cache d'index, lente mais sûre. **Notez** que ce paramétre ètait donnè en mèga-octets avant la version 4.0.3 et en octets depuis cette version.

`-O myisam_max_sort_file_size=#`	Ne pas utiliser la mèthode de tri rapide pour crèer l'index, si le fichier temporaire dèpasse la taille indiquèe ici. **Notez** que ce paramétre ètait donnè en mèga-octets avant la version 4.0.3 et en octets depuis cette version.
`-O bulk_insert_buffer_size=#`	Taille du cache d'arbre lors d'insertion massives. **Notez** que c'est une limite **par thread**!

La rèparation automatique est activèe si vous dèmarrez `mysqld` avec l'option `--myisam-recover=#`. Voir Section 4.1.1 [Command-line options], page 192. Lors de l'ouverture, la marque de fermeture de la table est vèrifièe, ou le compteur d'ouverture de la table n'est pas 0 et vous fonctionnez avec l'option `--skip-external-locking`. Si l'un des deux ci-dessus est vrai, la procèdure suivante s'applique :

- La table est vèrifièe.
- Si une erreur est trouvèe, une rèparation rapide est tentèe (avec tri et sans recrèation du fichier de donnèes).
- Si la rèparation rapide èchoue á cause d'une erreur dans le fichier de donnèes, (par exemple, une erreur de clè dupliquèe), nous essayons á nouveau, mais cette fois ci, avec recrèation du fichier de donnèes.
- Si cette rèparation èchoue, on essaie encore une fois avec la mèthode traditionnelle de rèparation (ècriture des lignes une á une, sans tri), qui doit rèparer la table de toutes ses erreurs, avec peu d'espace disque...

Si la rèparation n'a pas ètè capable de retrouver toutes les lignes d'une requête prècèdente, et que vous ne spècifiez pas l'option `FORCE` á `myisam-recover`, alors la correction automatique va s'arrêter avec l'erreur suivante dans le fichier d'erreurs :

```
Error: Couldn't repair table: test.g00pages
```

Dans ce cas, si vous utilisez l'option `FORCE`, vous obtiendrez le message suivant dans le fichier d'erreurs :

```
Warning: Found 344 of 354 rows when repairing ./test/g00pages
```

Notez que si vous exècutez une restauration automatique avec une option automatique de `BACKUP`, vous devriez avoir un script `cron` qui va automatiquement copier les fichiers avec l'extension '`tablename-datetime.BAK`' depuis le dossier de donnèes vers une solution de sauvegarde.

Voir Section 4.1.1 [Command-line options], page 192.

7.1.1 Espace requis pour les clefs

MySQL supporte plusieurs types d'index, mais le type normal est ISAM ou MyISAM. Ils utilisent un index B-tree, et vous pouvez avoir une approximation de la taille du fichier d'index en faisant la somme de `(longueur_clef+4)/0.67` pour toutes les clefs. (Cela est le pire des cas oú les clefs sont insèrèes dans l'ordre et qu'aucune n'est compressèe.

Les index de chaînes de caractéres sont compressès par rapport aux espaces. Si la premiére partie de l'index est une chaîne, son prèfixe sera aussi compressè. La compression des espaces rend le fichier d'index plus petit que ce que nous avions calculè prècèdement si la colonne chaîne posséde beaucoup d'espaces invisibles en dèbut et fin de chaînee ou est une

colonne `VARCHAR` qui n'est pas toujours pleinement utilisèe. La compression des prèfixes est utilisèe sur les clefs qui commencent par un chaîne de caractéres. La compression des prèfixes s'il y'a plusieurs chaînes avec des prèfixes identiques.

Dans les tables `MyISAM`, vous pouvez aussi compresser les nombres en spècifiant `PACK_KEYS=1` lors de la crèation de la table. Cela vous aidera lorsque vous aurez plusieurs clefs de types entier qui auront un prèfixe identique et que les nombres seront classè par ordre dècroissant des grands octets.

7.1.2 Formats de table `MyISAM`

`MyISAM` supporte 3 diffèrent types de tables. Deux des trois sont choisis automatiquement selon le type de colonne que vous utilisez. Le troisiéme, tables compressèes, ne peut être crèe qu'avec l'outil `myisampack`.

Quand vous crèez une table avec `CREATE` ou en modifiez la structure avec `ALTER` vous pouvez, pour les tables n'ayant pas de champs `BLOB` forcer le type de table en `DYNAMIC` ou `FIXED` avec l'option `ROW_FORMAT=#` des tables. Bientôt, vous pourrez compresser/dècompresser les tables en spècifiant `ROW_FORMAT=compressed | default` á `ALTER TABLE`. Voir Section 6.5.3 [CREATE TABLE], page 504.

7.1.2.1 Caractèristiques des tables statiques (taille fixèe)

Ceci est le format par dèfaut. Il est utilisè lorsque la table ne contient pas de colonnes de type `VARCHAR`, `BLOB`, ou `TEXT`.

Ce format est le plus simple et le plus sûr. C'est aussi le format sur disque le plus rapide. La vitesse vient de la facilitè avec laquelle les donnèes peuvent être trouvèes sur le disque. La recherche de quelque chose avec un index et un format statique est trés simple. Multipliez juste le nombre de lignes par la longueur des lignes.

De même, lors du scannage d'une table, il est trés facile de lire un nombre constant d'enregistrements avec chaque lecture du disque.

La sècuritè est mise en èvidence si votre ordinateur crashe lors de l'ècriture dans un fichier de taille fixèe MyISAM, dans ce cas, `myisamchk` peur facilement trouver oú commence et finit chaque ligne. Il peut donc retrouver tous les enregistrements á part celui dont l'ècriture a ètè interrompue. Notez qu'avec MySQL tous les index peuvent toujours être reconstruits :

- Toutes les colonnes `CHAR`, `NUMERIC`, et `DECIMAL` sont complètèes par des espaces jusqu'á atteindre la longueur totale de la colonne.
- Trés rapide.
- Facile á mettre en cache.
- Facile á reconstruire aprés un crash, car les enregistrements sont localisès dans des positions fixèes.
- N'a pas á être rèorganisè (avec `myisamchk`) sauf si un grand nombre de lignes est effacè et que vous voulez retourner l'espace libèrè au systéme d'exploitation.
- Requiére usuellement plus d'espace disque que les tables dynamiques.

7.1.2.2 Caractèristiques des tables á format de ligne dynamiques

Ce format est utilisè avec les tables qui contiennent des colonnes de type `VARCHAR`, `BLOB` ou `TEXT`, ou si la table a ètè crèèe avec l'option `ROW_FORMAT=dynamic`.

Ce format est un peu plus complexe, car chaque ligne doit avoir un entête pour indiquer sa longueur. Une ligne peut aussi être rèpartie sur plusieurs blocs, lorsqu'elle est agrandie lors d'une modification.

Vous pouvez utiliser la commande SQL `OPTIMIZE table` ou shell `myisamchk` pour dèfragmenter une table. Si vous avez des donnèes statiques que vous modifiez souvent dans la même table, avec des colonnes `VARCHAR` ou `BLOB`, il peut être une bonne idèe de placer des colonnes dans une autre table, pour èviter la fragmentation :

- Toutes les colonnes de type chaîne sont dynamiques (hormis celle qui sont de taille infèrieure á 4).
- Chaque ligne est prècèdèe d'un octet qui indique quelles sont les lignes vides (`''`, bit á 1) et celle qui le ne sont pas (bit á 0). Une colonne vide n'est pas la même choses qu'une colonne qui contient `NULL`. Si une colonne a une taille de zèro aprés avoir supprimè les espaces finaux, ou un nombre a une valeur de zèro, il est marquè dans cet octet, et la colonne sera ignorèe sur le disque. Les chaînes non vides sont sauvèes avec un octet de plus pour y stocker la taille.
- Ce format prend gènèralement moins de place que des tables á format fixe.
- Chaque ligne consomme autant d'espace que nècessaire. Si une ligne devient trop grande, elle sera coupèe en blocs et ècrites dans le fichier de donnèes. Cela engendre la fragmentation du fichier de donnèes.
- Si vous modifiez une ligne avec des informations qui excédent la capacitè courante de la ligne, la ligne sera fragmentèe. Dans ce cas, vous pouvez avoir á exècuter la commande `myisamchk -r` de temps en temps pour amèliorer les performances. Utilisez `myisamchk -ei tbl_name` pour obtenir des statistiques.
- Ce format de table n'est pas toujours facile á reconstituer aprés un crash, car une ligne peut êtr fragmentèe en de nombreux blocs, et un fragment peut manquer.
- La taille d'une ligne de format variable se calcule avec :

```
3
+ (nombre de colonnes + 7) / 8
+ (nombre de colonnes de tailles chars)
+ taille compactèe des colonnes numèriques
+ taille des chaînes
+ (nombre de colonne de valeur NULL + 7) / 8
```

  Il y a un aussi un supplèment de 6 octets pour chaque lien. Une ligne de format dynamique utilise un lien á chaque fois qu'une modification cause un agrandissement de la ligne. Chaque nouveau bloc liè fait au moins 20 octets, pour que le prochain agrandissement utilise aussi ce bloc. Si ce n'est pas le cas, un nouveau bloc sera liè, avec un autre coût de 6 octets. Vous pouvez vèrifier le nombre de liens dans une table avec la commande `myisamchk -ed`. Tous les liens sont supprimès avec la commande `myisamchk -r`.

7.1.2.3 Caractèristiques des tables compressèes

C'est un type en lecture seule qui est gènèrè avec l'outil optionnel `myisampack` (`pack_isam` pour les tables `ISAM`) :

- Toutes les distributions MySQL, même celles qui existaient avant que MySQL ne passe sous la licence GPL, peuvent lire des tables qui ont ètè comprèssèes avec `myisampack`.
- Les tables comprèssèes prennent trés peu d'espace disque. Cela amoindri l'espace requis ce qui est fort utile lors de l'utilisation de petits disques (comme les CD-ROM).
- Chaque ligne est compressèe sèparement (optimisation des accés). L'entête d'un enregistrement est fixè (1-3 octets) selon le plus grand enregistrement dans la table. Chaque colonne est compressèe diffèrement. Quelques un des types de compressions sont :
  - Il y'a usuellement une table Huffman diffèrente pour chaque colonne.
  - Compression des espaces en suffixe.
  - Compression des espaces en prèfixe.
  - Les nombres avec la valeur `0` sont stockès en utilisant 1 octet.
  - Si les valeurs dans une colonne de type entier ont un petit intervalle, la colonne est stockèe en utilisant le type le plus petit possible. Par exemple, une colonne `BIGINT` (8 octets) peut être stockè en tant que colonne `TINYINT` (1 octet) si toutes les valeurs sont entre `0` et `255`.
  - Si une colonne n'a qu'un petit èventail de valeurs, son type est changè en `ENUM`.
  - Une colonne peut utiliser une combinaison des compressions prècèdentes.
- Peut gèrer les enregistrements de tailles fixes ou variables.
- Peut être dècompressèe avec `myisamchk`.

7.1.3 Problémes avec les tables `MyISAM`

Le format de fichier que MySQL utilise pour stocker les donnèes a ètè testè á l'extrême, mais il y'a toujours des circonstances qui peuvent corrompre les tables d'une base de donnèes.

7.1.3.1 Tables `MyISAM` corrompues

Même si le format des tables MyISAM est relativement sûr (tous les changements sont ècrits avant que la requête SQL ne retourne quoi que ce soit), vous pouvez quand même vous trouver face á des tables corrompues si l'une des choses suivantes arrive :

- Le processus `mysqld` est tuè au milieu d'une ècriture.
- Arrêt inattendu de la machine (par exemple, coupure de courant).
- Un probléme matèriel.
- Vous utilisez un programme externe (comme myisamchk) sur une table active.
- Un bogue logiciel dans le code de MySQL ou de MyISAM.

Les symptômes typiques d'une table corrompue sont :

- Vous obtenez l'erreur `Incorrect key file for table: '...'. Try to repair it` pendant la sèlection de donnèes á partir de cette table.

- Les requêtes ne trouvent pas de lignes dans la table ou retournent des données incomplétes.

Vous pouvez vèrifier l'ètat d'une table avec la commande `CHECK TABLE`. Voir Section 4.4.4 [CHECK TABLE], page 244.

Vous pouvez rèparer une table corrompue avec `REPAIR TABLE`. Voir Section 4.4.5 [REPAIR TABLE], page 246. Vous pouvez aussi rèparer une table, lorsque `mysqld` ne fonctionne pas, avec la commande `myisamchk`. `myisamchk syntax`.

Si vos tables sont souvent corrompues, vous devez essayez de trouver d'oú vient le probléme ! Voir Section A.4.1 [Crashing], page 697.

Dans ce cas, la chose la plus importante á savoir est, si la table est corrompue, si le serveur `mysqld` s'est interrompu. (cela peut être facilement vèrifiè en regardant s'il y'a une entrèe rècente `restarted mysqld` dans le fichier d'erreurs de mysqld). Si ce n'est pas le cas, vous devez essayer d'effectuer une sèrie de tests. Voir Section E.1.6 [Reproduceable test case], page 820.

7.1.3.2 Clients is using or hasn't closed the table properly

Chaque fichier `MyISAM` '`.MYI`' posséde un compteur dans l'entête qui peut être utilisè pour savoir si une table a ètè fermèe Proprement.

Si vous obtenez l'avertissement suivant de la part de `CHECK TABLE` ou `myisamchk` :

```
# clients is using or hasn't closed the table properly
```

cela signifie que le compteur n'est plus synchrone. Cela ne signifie Pas que la table est corrompue, mais que vous devez au moins effectuer une vèrification sur la table pour vous assurer de son bon fonctionnement.

Le compteur fonctionne de la façon suivante :

- La premiére fois qu'une table est mise á jour dans MySQL, un compteur dans l'entête du fichier est incrèmentè.
- Le compteur ne change pas pour les mises á jours suivantes.
- Lors de la fermeture de la derniére instance d'une table (á cause d'un `FLUSH` ou qu'il n'y a plus de place dans le cache de la table) le compteur est decrèmentè si la table n'a pas ètè mise á jour.
- Lorsque vous rèparez la table ou vèrifiez quel est en bon ètat, le compteur est remis á zèro.
- Pour èviter les problémes d'interactions avec d'autres processus qui peuvent vèrifier la table, le compteur n'est pas dècrèmentè á la fermeture si sa valeur ètait zèro.

En d'autres termes, les seuls moyens d'obtenir ce genre d'erreur sont :

- Les tables `MyISAM` sont copiès sans `LOCK` et `FLUSH TABLES`.
- MySQL a plantè entre une mise á jour et la fermeture finale. (Notez que la table peut encore être bonne, vu que MySQL ècrit toujours pour tout entre deux requêtes.)
- quelqu'un a exècutè `myisamchk --recover` ou `myisamchk --update-state` sur une table qui ètait utilisèe par `mysqld`.

- Plusieurs serveurs `mysqld` utilisent la table et l'un d'eux a exècutè dessus un `REPAIR` ou un `CHECK` pendant qu'elle ètait utilisèe par un autre serveur. Dans ce cas lá, l'utilisation de `CHECK` n'est pas trés grave (même si vous obtiendrez des avertissements sur les autres serveurs), mais `REPAIR` doit être èvitèe vu qu'elle remplace actuellement le fichier de donnèes part un nouveau, ce qui n'est pas signalè aux autres serveurs.

7.2 Tables assemblèes `MERGE`

Les tables `MERGE` sont nouvelles depuis MySQL version 3.23.25. Le code est toujours en phase gamma, mais il est dèjá raisonnablement stable.

Une table `MERGE` (aussi connue sous le nom de `MRG_MyISAM`, ou table assemblèe) est un regroupement de tables `MyISAM` identiques, qui peuvent être utilisèes ensemble, comme une seule. Nous ne pouvez faire que des commandes `SELECT`, `DELETE` et `UPDATE` dans ces tables. Si vous effacez (avec `DROP`) la table `MERGE`, vous ne faîtes qu'annuler le rassemblement `MERGE`.

Notez que la commande `DELETE FROM merge_table` utilisèe sans la clause `WHERE` va seulement effacer le rassemblement de tables, et non pas les lignes dans les tables. Nous envisageons de corriger cela en version 4.1.

Avec des tables identiques, nous voulons dire que toutes les tables sont crèèes avec les mêmes colonnes et index. Vous ne pouvez pas rassembler des tables dans lesquelles les colonnes sont dèfinies diffèremment, n'ont pas exactement le même nombre de colonne, ou ont des index dans un autre ordre. Cependant, certaines des tables peuvent être compressèes avec `myisampack`. Voir Section 4.7.4 [`myisampack`], page 297.

Lorsque vous crèez une table `MERGE`, vous allez obtenir un fichier '`.frm`', de dèfinition de table, et un fichier de liste de tables '`.MRG`'. Le fichier '`.MRG`' contient simplement la liste des fichiers d'index (extension '`.MYI`') qui doivent être utilisès comme un seul et même. Toutes les tables utilisèes doivent être dans la même base que la table `MERGE` elle-même.

Pour le moment, vous avez simplement besoin des droits de `SELECT`, `UPDATE` et `DELETE` sur les tables que vous avez rassemblè dans la table `MERGE`.

Les tables `MERGE` peuvent vous aider dans les situations suivantes :

- Gèrer facilement un jeu de table de log. Par exemple, vous pourriez placer les donnèes de chaque mois dans un fichier sèparè, en compresser certains avec `myisampack` puis crèer une table `MERGE` pour les utiliser.
- Vous donner plus de vitesse. Vous pouvez rèpartir les grandes tables en lecture seule dans diffèrentes parties du disque. Une table `MERGE` bâtie de cette façon peut être plus rapide qu'une grosse table (vous pouvez aussi et bien sûr, utiliser un systéme RAID pour arriver aux mêmes avantages).
- Effectuer des recherches plus efficaces. Si vous savez exactement ce que vous recherchez, vous pouvez faire des recherches dans une seule des tables individuelles pour les recherches, et utiliser la table `MERGE` pour les autres opèratiosn. Vous pouvez même avoir de nombreuses tables `MERGE` actives, qui partagent les même fichiers.
- Des rèparations plus efficaces. Il est plus facile de rèparer les fichiers individuels qui sont rassemblès dans une table `MERGE` que de rèparer une grande table.
- Fusion instantannèe de plusieurs fichiers en un seul. Une table `MERGE` utilise les index des tables individuelles. Il n'y a pas besoin de gèrer un seul index. Cela rend les tables

MERGE trés rapides á faire ou dèfaire. Notez que vous devez spècifier les dèfinitiosn de clès lorsque vous crèeez la table MERGE!

- Si vous avez un jeu de table que vous rassemblez dans une grande á la demande ou pour un traitement batch, vous devriez utiliser une table MERGE. C'est bien plus rapide, et cela va vous faire èconomiser de l'espace disque.
- Contourner les limitations de taille du systéme d'exploitation.
- Vous pouvez crèer un alias ou un synomyne pour une table, en utilisant simplement MERGE sur une seule. Il n'y a pas de coûts particulier en performance (hormis quelques appels de fonctions indirects, et des `memcpy()` avant chaque lecture).

Les inconvènients des tables de type MERGE sont :

- Vous devez utiliser des tables MyISAM identiques pour faire une table MERGE.
- REPLACE ne fonctionne pas.
- MERGE utilise plus de pointeurs de fichiers. Si vous utilisez une table MERGE qui couvre 10 tables et que 10 utilisateurs l'utilisent, vous consommez 10*10 + 10 pointeurs de fichiers (10 fichiers de données, et 10 utilisateurs avec 10 fichiers d'index).
- Les lectures de clès sont plus lente. Lorsque vous faîtes une lecture sur une clè, le gestionaire MERGE doit faire une lectuer dans tous les fichiers d'index des tables sous-jacentes, pour vèrifier lequel est le plus proche de la valeur recherchèe. Si vous faîtes une lecture du type "lit le suivant", le gestionnaire de table assemblèe doit rechercher dans tous les buffers de clès pour la trouver. Uniquement lorsqu'un buffer clè est complet, doit il lire le prochain bloc. Cela rend l'accés aux clès MERGE bien plus lent que les recherches `eq_ref`, mais pas aussi lent que les recherches de type `ref`. Voir Section 5.2.1 [EXPLAIN], page 363.
- Vous ne pouvez pas faire de commande DROP TABLE, ALTER TABLE, DELETE FROM table_name sans clause WHERE, REPAIR TABLE, TRUNCATE TABLE, OPTIMIZE TABLE ou ANALYZE TABLE sur aucune des tables qui sont couvertes par une table MERGE qui est "ouverte". Si vous faîtes cela, la table MERGE peut toujours pointeur sur la table originale, et vous allez obtenir des rèsultats inattendus. Le plus simple palliatif est d'utiliser la commande FLUSH TABLES, pour s'assurer qu'aucune des tables MERGE ne reste ouverte.

Lorsque vous crèez une table MERGE, vous devez spècifier la liste des tables que vous allez utiliser, avec l'option `UNION(list-of-tables)`. Optionnellement, grâce á l'option INSERT_METHOD, vous pouvez spècifier si vous voulez que la table MERGE ajoute les nouvelles lignes dans la premiére ou la derniére table de l'UNION. Si vous spècifiez pas INSERT_METHOD ou si vous spècifiez NO, alors les commandes INSERT donnèe á la table MERGE retourneront une erreur.

L'exemple suivant vous montre comme utiliser les tables MERGE :

```
CREATE TABLE t1 (a INT AUTO_INCREMENT PRIMARY KEY, message CHAR(20));
CREATE TABLE t2 (a INT AUTO_INCREMENT PRIMARY KEY, message CHAR(20));
INSERT INTO t1 (message) VALUES ("Testing"),("table"),("t1");
INSERT INTO t2 (message) VALUES ("Testing"),("table"),("t2");
CREATE TABLE total (a INT AUTO_INCREMENT PRIMARY KEY, message CHAR(20))
            TYPE=MERGE UNION=(t1,t2) INSERT_METHOD=LAST;
```

Notez que vous pouvez aussi manipuler le fichier '.MRG' directement, hors du serveur MySQL :

```
shell> cd /mysql-data-directory/current-database
shell> ls -1 t1.MYI t2.MYI > total.MRG
shell> mysqladmin flush-tables
```

Maintenant, vous pouvez faire des commandes comme :

```
mysql> SELECT * FROM total;
+---+---------+
| a | message |
+---+---------+
| 1 | Testing |
| 2 | table   |
| 3 | t1      |
| 1 | Testing |
| 2 | table   |
| 3 | t2      |
+---+---------+
```

Notez que la colonne `a`, bien que dèclarèe `PRIMARY KEY`, n'est pas vraiment unique, car les tables `MERGE` ne peuvent pas garantir l'unicitè entre les tables sous -jacente á la `MyISAM`.

Pour redèfinir une table `MERGE`, vous pouvez faire ceci :

- `DROP` la table, puis recrèez la.
- Utilisez `ALTER TABLE table_name UNION(...)`
- Modifiez le fichier '`.MRG`' et utilisez la commande `FLUSH TABLE` sur la table `MERGE` et toutes les tables sous-jacentes, pour forcer le gestionnaire á relire la dèfinition.

7.2.1 Problémes avec les tables `MERGE`

Voici une liste des problémes connus avec les tables de type `MERGE` :

- Une table `MERGE` peut pas pas supporter de contrainte de type `UNIQUE` sur toute la table. Lorsque vous faites une insertion, les donnèes vont dans la premiére ou la derniére table (suivant la mèthode d'insertion `INSERT_METHOD=xxx`) et cette table `MyISAM` s'assure que les donnèes sont uniques, mais rien n'est fait pour vèrifier l'unicitè auprés des autres tables `MyISAM` tables.
- `DELETE FROM merge_table` utilisè sans clause `WHERE` va uniquement dètuire la table assemblèe, mais ne va pas toucher les tables sous-jacentes.
- `RENAME TABLE` utilisè sur une table de type `MERGE` peut corrompre la table. Cela sera corrgiè en MySQL 4.0.x.
- La crèation d'une table de type `MERGE` ne vèrifie pas si les tables sous-jacentes sont comptabiles. Si vous utilisez une table `MERGE` de cette façon, vous devriez rencontrer des problémes trés ètranges.
- Si vous utilisez la commande `ALTER TABLE` pour ajouter un index de type `UNIQUE` á une table qui est utilisèe dans une table assemblèe `MERGE`, puis que vous utilisez `ALTER TABLE` pour ajouter un index normal dans la table `MERGE`, l'ordre des clè sera diffèrent suivant les tables, si jamais il y avait une vieille clè non unique. Ceci est dû au fait que `ALTER TABLE` place les clès `UNIQUE` avant les clès normales, pour être capable de dètecter les doublons le plus tôt possible.

- L'optimiseur d'intervalle ne peut pas encore utiliser les tables `MERGE` efficacement, et il produit parfois des jointures sub-optimales. Cela sera corrigè en MySQL 4.0.x.
- `DROP TABLE` sur une table qui est utilisè par une table `MERGE` ne fonctionne pas sous Windows car le gestionnaire de `MERGE` garde les connexions vers les tables cachèes sous la couche MySQL. Comme Windows ne vous permet pas d'effacer une table qui est ouverte, vous devez d'abord fermer toute les tables `MERGE` (avec la commande `FLUSH TABLES`) ou effacer la table `MERGE` avant de pouvoir effacer la table dèsirèe. Nous allons corriger lorsque nous introduirons les vues. `VIEW`s.

7.3 Tables `ISAM`

Vous pouvez aussi utiliser le type de tables dèsapprouvè `ISAM`. Il disparaîtra bientôt (probablement dans MySQL 5.0) car `MyISAM` est une meilleure implèmentation de la même chose. `ISAM` utilise un index `B-tree`. L'index est stockè dans un fichier portant l'extension '`.ISM`', et les donnèes sont enregistrèes dans un fichier avec l'extension '`.ISD`'. Vous pouvez vèrifier/rèparer les tables `ISAM` avec l'utilitaire `isamchk`. Voir Section 4.4.6.7 [Crash recovery], page 253.

`ISAM` posséde les fonctionnalitès/propriètès suivantes :

- Clefs compressèes et de tailles fixes
- Enregistrements de taille fixèe ou dynamique
- 16 clefs avec 16 parties de clefs/clefs
- Taille maximale de la clef 256 (dèfaut)
- Les donnèes sont enregistrèes au format machine; c'est rapide, mais c'est dèpendant de la machine/systéme d'exploitation.

La plupart des choses vraies pour les tables `MyISAM` le sont pour les tables `ISAM`. Voir Section 7.1 [`MyISAM` tables], page 532. La diffèrence majeure comparèes aux tables `MyISAM` sont :

- Les tables `ISAM` ne sont pas binairement portables entre les plate-formes/systémes d'exploitations.
- Ne peut gèrer les tables > 4G.
- Ne supporte que la compression des prèfixes sur les chaînes de caractéres.
- Limites de clefs plus petites.
- Les tables dynamiques sont plus fragmentèes.
- Les tables sont compressèes avec `pack_isam` plutôt qu'avec `myisampack`.

Si vous voulez convertir vos tables `ISAM` en `MyISAM` pour pouvoir utiliser des utilitaires tels que `mysqlcheck`, utilisez la commande `ALTER TABLE` :

```
mysql> ALTER TABLE nom_de_table TYPE = MYISAM;
```

Les versions embarquèes de MySQL ne supportent pas les tables `ISAM`.

7.4 Tables HEAP

Les tables HEAP utilisent un index de hachage, et sont stockèes en mèmoire. Elles sont trés rapides, mais si MySQL crashe, vous perdrez toutes vos donnèes. Les tables HEAP sont trés pratiques pour être des tables temporaires!

Les tables HEAP internes MySQL utilisent des hachages 100% dynamiques sans zones de dèbordement. Il n'y a pas d'espace nècessaire pour des listes libres. Les tables HEAP n'ont aussi aucun probléme d'effacement et d'insertions, qui sont le lot commun des tables de hachage :

```
mysql> CREATE TABLE test TYPE=HEAP SELECT ip,SUM(downloads) AS down
    ->                   FROM log_table GROUP BY ip;
mysql> SELECT COUNT(ip),AVG(down) FROM test;
mysql> DROP TABLE test;
```

Voici quelques conseils concernant les tables HEAP :

- Vous devez toujours penser á spècifier le nombre maximal de lignes de la table avec MAX_ROWS dans la commande de crèation CREATE pour vous assurer de ne pas accidentellement consommer toute la mèmoire.
- Les index ne peuvent être utilisès qu'avec les opèrateurs = et <=> (mais ils sont trés rapides).
- Les tables HEAP ne peuvent utiliser que des clès entiéres pour rechercher une ligne. Par comparaisons, les tables MyISAM peuvent utiliser un prèfixe de clè pour trouver des lignes.
- Les tables HEAP utilisent un format de ligne fixe.
- HEAP ne supporte pas les colonnes de type BLOB/TEXT.
- HEAP ne supporte pas les colonnes de type AUTO_INCREMENT.
- Avant MySQL 4.0.2, HEAP ne supportait les index sur les valeurs NULL.
- Vous pouvez utiliser des clès non-uniques avec les tables HEAP (ce qui n'est pas courant pour les tables de hachage).
- Les tables HEAP sont partagèes entre tous les clients (comme une autre table).
- Vous ne pouvez pas recherche la prochaine ligne (c'est á dire, utiliser l'index pour optimiser la clause ORDER BY).
- Les donnèes pour les tables HEAP sont allouès par petits blocs. Les tables sont 100% dynamiques (en insertion). Aucune zone de dèbordement ou d'espace de clè supplèmentaire n'est nècessaire. Les lignes effacèes sont placèes dans une liste, prêtes á être rèutilisèes.
- Vous avez besoin de suffisamment de mèmoire pour accepter toutes les tables HEAP que vous allez utiliser simultanèment.
- Pour libèrer de la mèmoire, vous devez exècuter la commande DELETE FROM heap_table, TRUNCATE heap_table ou DROP TABLE heap_table.
- MySQL ne peut pas calculer approximativement le nombre de lignes entre deux valeurs (ce qui est utilisè par l'optimiseur pour savoir quelle mèthode utiliser). Cela peut affecter certaines requêtes, si vous changez une table MyISAM en table HEAP.

- Pour vous assurer que vous ne faîtes rien de trop tèmèraire, vous ne pouvez pas crèer de table `HEAP` plus grande que `max_heap_table_size`.

La mèmoire nècessaire pour les tables `HEAP` sont :

```
SUM_OVER_ALL_KEYS(max_length_of_key + sizeof(char*) * 2)
+ ALIGN(length_of_row+1, sizeof(char*))
```

`sizeof(char*)` vaut 4 sur les machines 32 bits et 8 sur une machine 64 bits.

7.5 Tables `InnoDB`

7.5.1 Prèsentation des tables InnoDB

InnoDB fournit á MySQL un gestionnaire de table transactionnelle (compatible `ACID`), avec validation (commits), annulations (rollback) et capacitès de restauration aprés crash. InnoDB utilise un verrouillage de lignes, et fournit des lectures cohèrentes comme Oracle, sans verrous. Ces fonctionnalitès accroissent les possibilitès d'utilisation simultanèes des tables, et les performances. Il n'y a pas de probléme de queue de verrous avec InnoDB, car les verrous de lignes utilisent trés peu de place. Les tables InnoDB sont les premiéres tables MySQL qui supportent les contraintes de clès ètrangéres (`FOREIGN KEY`).

InnoDB a ètè conçu pour maximiser les performances lors du traitement de grandes quantitès de donnèes. Son efficacitè processeur n'est ègalèe par aucun autre moteur de base de donnèes.

Techniquement, InnoDB est un gestionnaire de table placè sous MySQL. InnoDB dispose de son propre buffer pour mettre en cache les donnèes et les index en mèmoire centrale. InnoDB stocke les tables et index dans un espace de table, qui peut être rèparti dans plusieurs fichiers. Ceci différe des tables comme, par exemple, MyISAM oú chaque table est stockèe dans un fichier diffèrent. Les tables InnoDB peuvent prendre n'importe quelle taille, même sur les systémes d'exploitation dont la limite est de 2 Go par fichier.

Vous pouvez trouver les derniéres informations sur InnoDB á `http://www.innodb.com/`. La derniére version du manuel InnoDB est toujours disponible la-bas, et vous pouvez aussi commander des licences commerciales et du support InnoDB.

InnoDB est actuellement (Octobre 2001) utilisè en production dans plusieurs sites oú de grandes capacitès de stockages et des performances accrues sont nècessaires. Le fameux site web Slashdot.org utilise InnoDB. Mytrix, Inc. stocke plus de 1 To de donnèes dans une base InnoDB, et un autre site gére une moyenne de 800 insertions/modifications par secondes avec InnoDB.

Les tables InnoDB sont inclues dans la distribution source de MySQL depuis la version 3.23.34a et sont activèes dans le binaire MySQL-Max. Pour Windows, les binaires de MySQL-Max sont disponibles dans la distribution standard.

Si vous avez tèlèchargè une version binaire de MySQL qui inclut le support de InnoDB, suivez simplement les instructions du manuel MySQL pour installer une version binaire. Is vous avez dèjá la version MySQL-3.23 installèe, alors le moyen de plus simple est d'installer le serveur '`mysqld`' avec l'exècutable correspondant de la distribution -Max. MySQL et

MySQL -Max ne différent que par leur exècutable. Voir Section 2.2.8 [Installing binary], page 82. Voir Section 4.7.5 [mysqld-max], page 304.

Pour compiler MySQL avec le support InnoDB, tèlèchargez MySQL-3.23.34a ou plus rècent depuis `http://www.mysql.com/` et configurez MySQL avec l'option `--with-innodb`. Voyez le manuel MySQL sur l'installation d'une distribution source. Voir Section 2.3 [Installing source], page 84.

```
cd /chemin/vers/source/de/mysql-3.23.37
./configure --with-innodb
```

Pour utiliser InnoDB, vous devez spècifier les options de dèmarrage InnoDB dans le fichier '`my.cnf`' ou '`my.ini`'. La mèthode minimale pour modifier ces options est d'ajouter á la section `[mysqld]`, la ligne

```
innodb_data_file_path=ibdata:30M
```

mais pour obtenir de meilleures performances, vous devez spècifier les options telles que recommandè. Voir Section 7.5.2 [InnoDB start], page 545.

InnoDB est sous licence GNU GPL License Version 2 (de Juin 1991). Dans la distribution source de MySQL, InnoDB apparaît comme un sous dossier.

7.5.2 Options de dèmarrage InnoDB

Pour utiliser les tables InnoDB en MySQL-Max-3.23, vous devez spècifier des paramétres de configuration dans la section `[mysqld]` du fichier de configuration '`my.cnf`', ou optionnelle, dans le fichier '`my.ini`' sous Windows.

Au minimum, en 3.23, vous devez spècifier `innodb_data_file_path` pour spècifier les noms et tailles de fichiers de donnèes. Si vous dècidez de ne pas mentionner `innodb_data_home_dir` dans '`my.cnf`', le comportement par dèfaut est de crèer ces fichiers dans le dossier de donnèes de MySQL. Si vous ne spècifiez pas `innodb_data_home_dir` sous la forme d'une chaîne vide, vous pouvez donner un chemin absolu jusqu'au stockage de vos donnèes dans `innodb_data_file_path`. En MySQL 4.0, vous n'avez même pas á spècifier l'option `innodb_data_file_path` : le comportement par dèfaut est de crèer un fichier de donnèes auto-croissant de 10 Mo appelè '`ibdata1`' dans le dossier de donnèes de MySQL. En MySQL 4.0.0 et 4.0.1, le fichier de donnèes ètait de 64 Mo et de taille fixe.

Si vous ne voulez pas utiliser les tables `InnoDB`, vous pouvez ajouter l'option `skip-innodb` dans le fichier d'options de MySQL.

Mais pour obtenir de bonnes performances, vous devez explicitement choisir les paramétres InnoDB listès dans les exemples suivants :

Depuis les versions 3.23.50 et 4.0.2, InnoDB fait que le dernier fichier spècifiè dans l'option `innodb_data_file_path` peut être auto-croissant (**auto-extending**). La syntaxe pour la ligne `innodb_data_file_path` est alors la suivante :

```
pathtodatafile:sizespecification;pathtodatafile:sizespecification;...
...  ;pathtodatafile:sizespecification[:autoextend[:max:sizespecification]]
```

Si vous spècifiez le dernier fichier avec l'option `autoextend`, InnoDB va augmenter la taille du dernier fichier de donnèes jusqu'á ce qu'il n'y ait plus de place dans l'espace de table. Les incrèments se feront par bloc de 8 Mo. Par exemple :

```
innodb_data_home_dir =
innodb_data_file_path = /ibdata/ibdata1:100M:autoextend
```

indique á InnoDB de crèer un fichier de donnèes unique, de taille initiale de 100 Mo et qui sera agrandi de 8Mo jusqu'á ce qu'il n'y ait plus de place. Si le disque se remplit, vous placerez le prochain fichier de donnèes sur un autre disque. Vous devez alors regarder la taille du fichier de donnèes '`ibdata1`', arrondir sa taille au Mo prècèdent (multiple de 1024 * 1024 octets (= 1 Mo)) et spècifier la taille du fichier '`ibdata1`' explicitement dans l'option `innodb_data_file_path`.

Aprés cela, vous pouvez spècifier un autre fichier de donnèes :

```
innodb_data_home_dir =
innodb_data_file_path = /ibdata/ibdata1:988M;/disk2/ibdata2:50M:autoextend
```

Soyez prudents avec les systémes de fichiers oú la taille maximale de fichier est de 2 Go! InnoDB n'est pas capable de dètecter la taille maximale de fichier pour votre systéme d'exploitation. Sur d'autres systémes, vous devrez spècifier la taille maximale du fichier :

```
innodb_data_home_dir =
innodb_data_file_path = /ibdata/ibdata1:100M:autoextend:max:2000M
```

Exemple de fichier '`my.cnf`' simple Supposons que vous avez un serveur avec 128 Mo de RAM et un disque dur. Voici un exemple de configuration de fichier '`my.cnf`' ou '`my.ini`' pour InnoDB. Nous supposons que vous exècutez MySQL-Max-3.23.50 ou plus rècent, ou MySQL-4.0.2 ou plus rècent.

Cet exemple devrait convenir á une majoritè d'utilisateurs, Unix et Windows, qui ne souhaitent pas rèpartir leur fichiers de donnèes InnoDB et leurs logs sur plusieurs disques. Cette configuration crèe un fichier de donnèes auto-croissant, appelè '`ibdata1`' et deux fichiers de log InnoDB '`ib_logfile0`' et '`ib_logfile1`' dans le dossier de donnèes MySQL (typiquement '`/mysql/data`'). De plus, le petit fichier d'archive InnoDB '`ib_arch_log_0000000000`' sera placè dans `datadir`.

```
[mysqld]
# Vous pouvez placer d'autres options MYSQL ici
# ...
#                                   Le fichier de donnèes doivent contenir
#                                   vos donnèes et index.
#                                   Assurez vous que vous avez l'espace disque
#                                   nècessaire.
innodb_data_file_path = ibdata1:10M:autoextend
#                                   Utilisez un buffer de taille
#                                   50 á 80 % de votre mèmoire serveur
set-variable = innodb_buffer_pool_size=70M
set-variable = innodb_additional_mem_pool_size=10M
#                                   Utiisez un fichier de log de taille
#                                   25 % du buffer mèmoire
set-variable = innodb_log_file_size=20M
set-variable = innodb_log_buffer_size=8M
#                                   Utilisez  ..flush_log_at_trx_commit
#                                   á 0 si vous pouvez accepter de perdre
#                                   quelques transactions
```

```
innodb_flush_log_at_trx_commit=1
```

Vèrifiez que le serveur MySQL a les droits de crèer ces fichiers dans le `datadir`.

Notez que le fichier de donnèes doit être infèrieure á 2Go sur certains systémes d'exploitation. La taille combinèe des fichiers de log doit être infèrieure á 4Go. La taille combinèe des fichiers de donnèes doit être infèrieure á 10Go.

Lorsque vous crèez pour la premiére fois une base de donnèes InnoDB, il est mieux de lancer le serveur depuis la commande en ligne. InnoDB va afficher des informations sur la crèation de la base, et vous verrez commence ça se passe. Voyez la section plus bas, pour une illustration. Par exemple, sous Windows, vous pouvez dèmarrer '`mysqld-max.exe`' avec :

```
chemin-jusqu-a-mysqld>mysqld-max --console
```

Oz mettre le fichier '`my.cnf`' ou '`my.ini`' sous Windows? Les régles sous Windows sont les suivantes :

- Un seul des deux fichiers '`my.cnf`' ou '`my.ini`' doit être crèè.
- The '`my.cnf`' doit être placè dans le dossier racine du disque '`C:`'.
- Le fichier '`my.ini`' doit être placè dans le dossier `WINDIR`, e.g, '`C:\WINDOWS`' ou '`C:\WINNT`'. Vous pouvez utiliser la commande `SET` de MS-DOS pour afficher la valeur de `WINDIR`.
- Si votre PC utilise un gestionnaire de dèmarrage oú le '`C:`' n'est pas votre disque de dèmarrage, alors votre seule option est d'utilise le fichier '`my.ini`'.

Oz placer les fichiers d'options sous Unix? Sous Unix, '`mysqld`' lit les options dans les fichiers suivants, si ils existent, et dans cet ordre :

- '`/etc/my.cnf`' Options globales.
- '`COMPILATION_DATADIR/my.cnf`' Options spècifiques au serveur.
- '`defaults-extra-file`' Le fichier spècifiè avec `--defaults-extra-file=...`.
- '`~/.my.cnf`' User-specific options.

'`COMPILATION_DATADIR`' est le dossier de donnèes de MySQL qui a ètè spècifiè lors de l'utilisation du script `./configure`, avant la compilation de '`mysqld`'. (typiquement, '`/usr/local/mysql/data`' pour une installation binaire, ou '`/usr/local/var`' pour une installation source).

Si vous n'êtes pas sûr des chemins oú '`mysqld`' lit les donnèes '`my.cnf`' et '`my.ini`', vous pouvez indiquer le chemin, avec la premiére option du serveur : `mysqld --defaults-file=your_path_to_my_cnf`.

InnoDB forme le chemin de dossier en concatènant `innodb_data_home_dir` avec les noms de fichiers, ou le chemin de `innodb_data_file_path`, en ajoutant les slash nècessaires. Si le mot clè `innodb_data_home_dir` n'est pas mentionnè dans '`my.cnf`', la valeur par dèfaut est 'point' '`./`', ce qui signifie le dossier de donnèes de MySQL.

Exemple de fichier '`my.cnf`' avancè Supposons que vous avez un serveur Linux avec 2 Go de RAM et trois disques de 60 Go (situès dans les dossiers '`/`', '`/dr2`' et '`/dr3`'. Voici ci-dessous un exemple de configuration possible pour '`my.cnf`', de InnoDB.

Notez que InnoDB ne crèe pas de dossier : vous devez le crèer vous même. Utilisez la commande Unix ou MS-DOS `mkdir` pour crèer les rèpertoires de donnèes et de logs.

```
[mysqld]
# Vous pouvez placer d'autres options MYSQL ici
# ...
innodb_data_home_dir =
#                                   Le fichier de données doivent contenir
#                                   vos données et index.
#                                   Assurez vous que vous avez l'espace disque
#                                   nècessaire.
innodb_data_file_path = /ibdata/ibdata1:2000M;/dr2/ibdata/ibdata2:2000M:autoextend
#                                   Utilisez un buffer de taille
#                                   50 á 80 % de votre mèmoire serveur
#                                   mais assurez vous sous Linux que l'utilisation
#                                   totale est infèrieure á 2Go
set-variable = innodb_buffer_pool_size=1G
set-variable = innodb_additional_mem_pool_size=20M
innodb_log_group_home_dir = /dr3/iblogs
#                                   .._log_arch_dir doit être le même que
#                                   .._log_group_home_dir
innodb_log_arch_dir = /dr3/iblogs
set-variable = innodb_log_files_in_group=3
#                                   Utiisez un fichier de log de taille
#                                   15 % du buffer mèmoire
set-variable = innodb_log_file_size=150M
set-variable = innodb_log_buffer_size=8M
#                                   Utilisez  ..flush_log_at_trx_commit
#                                   á 0 si vous pouvez accepter de perdre
#                                   quelques transactions
innodb_flush_log_at_trx_commit=1
set-variable = innodb_lock_wait_timeout=50
#innodb_flush_method=fdatasync
#set-variable = innodb_thread_concurrency=5
```

Notez que nous avons placè deux fichier de donnèes sur des disques diffèrents. InnoDB va remplir l'espace de tables jusqu'au maximum. Dans certains cas, les performances seront amèliorèes si les donnèes ne sont pas toutes placèes sur le même disque physique. Placer les fichiers de log dans des disques sèparès est souvent une bonne chose. Vous pouvez aussi utiliser des **partitions de disques brutes** (raw devices) comme fichier de donnèes. Sur certains Unix, cela va accèlèrer les accés disques. Voyez le manuel de InnoDB, pour savoir comment les spècifier dans '`my.cnf`'.

Attention : en Linux x86, vous devez être trés prudent, et **ne pas utiliser trop de mèmoire**. `glibc` va autoriser les processus á dèpasser la pile de thread, et votre systéme va crasher. Cela reprèsente un risque rèel si la valeur de

```
innodb_buffer_pool_size + key_buffer +
max_connections * (sort_buffer + read_buffer_size) + max_connections * 2 MB
```

est proche de 2 Go ou excéde 2 Go. Chaque thread va utiliser une pile (souvent 2Mo, mais les exècutables MySQL uniquement 256 ko) et dans le pire des scènarios, `sort_buffer + read_buffer_size` de mèmoire supplèmentaire.

Comment optimiser d'autres paramétres du serveur 'mysqld'? Les valeurs qui conviennent á la majoritè des utilisateurs sont :

```
skip-locking
set-variable = max_connections=200
set-variable = read_buffer_size=1M
set-variable = sort_buffer=1M
#                                  key_buffer vaut de 5 á 50%
#                                  de la RAM disponible, suivant l'utilisation des
#                                  tables MyISAM, mais garder
#                                  key_buffer + InnoDB
#                                  en deça de < 80% de votre RAM
set-variable = key_buffer=...
```

Notez que certains paramétres sont donnèes au format numèrique dans 'my.cnf' : `set-variable = innodb... = 123`, d'autres (chaînes et boolèens) sont donnèes dans un autre format : `innodb_... = ...` .

Les significations des paramétres de configuration sont les suivantes :

Option	Description
`innodb_data_home_dir`	La partie commune du chemin de tous les fichiers de donnèes InnoDB. Si vous ne mentionnez pas cette option dans 'my.cnf', la valeur par dèfaut sera celle du dossier de donnèes MySQL. Vous pouvez aussi spècifier une chaîne vide, et dans ce cas, les chemins spècifiès dans `innodb_data_file_path` seront des chemins absolus.
`innodb_data_file_path`	Chemin individuel vers les fichiers de donnèes, et leur taill. Le chemin complet de chaque fichier de donnèes est crèè en concatènant `innodb_data_home_dir` avec les chemins spècifiès ici. La taille du fichier est spècifiée en mèga-octets, ce qui explique la prèsence du 'M' aprés les spècifications ci-dessus. Depuis la version 3.23.44, vous pouvez donner au fichier une taille supèrieure á 4 Go sur les systémes d'exploitation qui acceptent les gros fichiers. Sur certains systémes, la taille doit être infèrieure á 2 Go. La somme des tailles des fichiers doit faire au moins 10 Mo.
`innodb_mirrored_log_groups`	Nombre de copies identiques de groupe de log que nous conservons. Actuellement, cette valeur doit être au minimum de 1.
`innodb_log_group_home_dir`	Le dossier pour les fichiers de logs.
`innodb_log_files_in_group`	Nombre de fichier de logs dans un groupe. InnoDB ècrit les logs de maniére circulaire. Une valeur de 3 est recommandèe ici.

`innodb_log_file_size`	Taille de chaque fichier de log dans un groupe de log, exprimè en mèga-octets. Les valeurs pratiques vont de 1Mo á une fraction de la taille du buffer de log (1 / le nombre de logs, en fait). Plus la taille est grande, moins de points de contrôles seront utilisès, rèduisant les accés disques. La taille combinèe des logs doit être infèrieure á 4 Go sur les systémes 32 bits.
`innodb_log_buffer_size`	La taille du buffer que InnoDB utilise pour ècrire les log dans les fichiers de logs, sur le disque. Les valeurs utiles vont de 1 Mo á 8 Mo. Un grand buffer de log permet aux grandes transactions de s'exècuter sans avoir á ècrire de donnèes dans le fichier de log jusqu'á la validation. Par consèquent, si vous avez de grandes transactions, augmenter cette taille va rèduire les accés disques.
`innodb_flush_log_at_trx_commit`	Normalement, cette option vaut 1, ce qui signifie que lors de la validation de la transaction, les logs sont ècrits sur le disque, et les modifications faites par la transaction deviennent permanentes, et survivront un crash de base. Si vous souhaitez rèduire la sècuritè de vos donnèes, et que vous exècutez de petites transactions, vous pouvez donner une valeur de 0 á cette option, pour rèduire les accés disques.
`innodb_log_arch_dir`	Le dossier oú les logs complètès doivent être archivès, si nous utilisons l'archivage de logs. La valeur de ce paramétre doit être actuellement la même que la valeur de `innodb_log_group_home_dir`.
`innodb_log_archive`	Cette valeur doit être actuellement de 0. Au moment de la restauration de donnèes á partir d'une sauvegarde, á l'aide des log binaires de MySQL, il n'y a actuellement pas besoin d'archiver les fichiers de log InnoDB.
`innodb_buffer_pool_size`	La taille de buffer mèmoire que InnoDB utiliser pour mettre en cache les donnèes et les index de tables. Plus cette valeur est grand, et moins vous ferez d'accés disques. Sur un serveur dèdiès, vous pouvez monter cette valeur jusqu'á 80% de la mèmoire physique de la machine. Ne lui donnez pas une valeur trop grande, car cela peut engendrer l'utilisation de mèmoire sur le disque par votre serveur.
`innodb_additional_mem_pool_size`	La taille du buffer mèmoire d'InnoDB, pour ses dictionnaires d'informations, et ses structures internes de donnèes. Une valeur pratique est 2Mo, mais plus vous aurez de tables dans votre application, plus vous devrez augmenter cette valeur. Si InnoDB est á court de mèmoire, il va allouer de la mèmoire auprés du systéme, et ècrire des messages dans le fichier de logs MySQL.

`innodb_file_io_threads`	Nombre de pointeurs de fichier de InnoDB. Normalement, cette valeur doit être de 4, mais sur des disques Windows, les accés peuvent être amèliorès en augmentant cette valeur.
`innodb_lock_wait_timeout`	Le dèlai d'expiration des transactions InnoDB, en cas de blocage de verrou, avant d'annuler. InnoDB dètecte automatiquement les blocages de verrous et annule alors les transactions. Si vous utilisez la commande `LOCK TABLES`, ou un autre gestionnaire de table transactionnelles que InnoDB dans la même transaction, un blocage de verrou peut survenir, et InnoDB ne pourra pas le dètecter. Ce dèlai est donc pratique pour rèsoudre ces situations.
`innodb_flush_method`	(Disponible depuis 3.23.40 et plus rècent) La valeur par dèfaut pour cette option est `fdatasync`. Une autre option est `O_DSYNC`.

7.5.3 Crèer des bases InnoDB

Supposons que vous avez installè MySQL et que vous avez èditè le fichier '`my.cnf`' de façon á ce qu'il contiennent les paramétres de configuration nècessaires de InnoDB. Avant de dèmarrer MySQL, vous devez vèrifier que les dossiers que vous avez spècifiè pour les fichiers de donnèes InnoDB et les fichiers de logs existent, et que vous avez des accés suffisants dans ces dossiers. InnoDB ne peut pas crèer de dossiers, uniquement des fichiers. Vèrifiez aussi que vous avez d'espace disque pour les donnèes et les logs.

Lorsque vous dèmarrez MySQL, InnoDB va commencer á crèer vos fichiers de donnèes et vos fichiers de log. InnDB va afficher ceci :

```
~/mysqlm/sql > mysqld
InnoDB: The first specified datafile /home/heikki/data/ibdata1
did not exist:
InnoDB: a new database to be created!
InnoDB: Setting file /home/heikki/data/ibdata1 size to 134217728
InnoDB: Database physically writes the file full: wait...
InnoDB: datafile /home/heikki/data/ibdata2 did not exist:
new to be created
InnoDB: Setting file /home/heikki/data/ibdata2 size to 262144000
InnoDB: Database physically writes the file full: wait...
InnoDB: Log file /home/heikki/data/logs/ib_logfile0 did not exist:
new to be created
InnoDB: Setting log file /home/heikki/data/logs/ib_logfile0 size to 5242880
InnoDB: Log file /home/heikki/data/logs/ib_logfile1 did not exist:
new to be created
InnoDB: Setting log file /home/heikki/data/logs/ib_logfile1 size to 5242880
InnoDB: Log file /home/heikki/data/logs/ib_logfile2 did not exist:
new to be created
InnoDB: Setting log file /home/heikki/data/logs/ib_logfile2 size to 5242880
InnoDB: Started
mysqld: ready for connections
```

Une nouvelle base de donnèes InnoDB a ètè crèèe. Vous pouvez vous connecter au serveur MySQL avec votre client MySQL habituel, comme `mysql`. Lorsque vous arrêtez le serveur MySQL avec '`mysqladmin shutdown`', InnoDB va afficher :

```
010321 18:33:34  mysqld: Normal shutdown
010321 18:33:34  mysqld: Shutdown Complete
InnoDB: Starting shutdown...
InnoDB: Shutdown completed
```

Vous pouvez observer vos fichiers de donnèes et de logs, et vous apercevrez les fichiers crèès. Le dossier de log va aussi contenir un petit fichier appelè '`ib_arch_log_0000000000`'. Ce fichier est le rèsultat de la crèation de base, á partir duquel InnoDB a dèsactivè l'archivage des logs. Lorsque MySQL va être redèmarrè, l'affichage sera :

```
~/mysqlm/sql > mysqld
InnoDB: Started
mysqld: ready for connections
```

7.5.3.1 Si quelque chose se passe mal á la crèation de la base de donnèes

Si InnoDB renvoie une erreur de systéme d'exploitation lors d'une opèration sur fichier, habituellement le probléme est l'un des suivants :

- Vous n'avez pas crèe les dossiers de donnèes ou de logs InnoDB.
- '`mysqld`' n'a pas le droit de crèer des fichiers dans ces dossiers.
- '`mysqld`' ne lit pas le bon fichier '`my.cnf`' ou '`my.ini`', et donc ne voit pas les options que vous spècifiez.
- Le disque ou l'espace disque allouè est plein.
- Vous avez crèe un sous-dossier dont le nom est le même que celui d'un fichier de donnèes que vous avez spècifiè.
- Il y'a une erreur de syntaxe dans `innodb_data_home_dir` ou `innodb_data_file_path`.

Si quelque chose se passe mal lors de la crèation d'une base de donnèes InnoDB, vous devez effacer tous les fichiers crèès par InnoDB. Cela inclut tous les fichiers de donnèes, tous les journaux, les archives. Dans le cas oú vous avez dèjá crèes des tables InnoDB, effacez aussi les fichiers '`.frm`' concernès dans le dossier de donnèes de MySQL. Vous pourrez alors essayer une nouvelle crèation de base de donnèes InnoDB.

7.5.4 Crèer des tables InnoDB

Supposons que vous avez dèmarrè le client MySQL avec la commande `mysql test`. Pour crèer une table au format InnoDB vous devez spècifier le type `TYPE = InnoDB` lors de la crèation de table, dans la commande SQL :

```
CREATE TABLE CUSTOMER (A INT, B CHAR (20), INDEX (A)) TYPE = InnoDB;
```

Cette commande SQL va crèer une table et un index sur la colonne `A` dans la base InnoDB constituèe par les fichiers de donnèes que vous avez spècifiè dans le fichier de configuration '`my.cnf`'. De plus, MySQL va crèer un fichier '`CUSTOMER.frm`' dans le dossier de donnèes

de MySQL ‘test’. En interne, InnoDB va ajouter une entrèe dans son propre dictionnaire de donnèes une entrèe pour la table ’test/CUSTOMER’. De cette façon, vous pouvez crèer plusieurs table avec le même nom de CUSTOMER, mais dans d’autres bases MySQL, et les noms de seront pas en conflit avec InnoDB.

Vous pouvez demander la quantitè d’espace disponible dans l’espace de tables InnoDB avec la commande de statut de MySQL pour toutes les tables de type TYPE = InnoDB. La quantitè d’espace disponible apparait dans la section de commentaire de la commande SHOW. Par exemple :

```
SHOW TABLE STATUS FROM test LIKE ’CUSTOMER’
```

Notez que les statistiques que SHOW vous donne sur les tables InnoDB ne sont que des approximations : elles sont utilisèes pour les optimisations SQL par MySQL. Les tailles rèservèes d’index et de table, exprimèes en octets, sont prècises.

7.5.4.1 Convertir une table MyISAM en InnoDB

InnoDB n’a pas d’optimisation particuliére pour la crèation d’un fichier externe d’index. Donc, exporter pour rèimporter les donnèes n’apporte pas de gain de performances. La mèthode la plus rapide pour donner á une table le format InnoDB est de faire les insertions directement dans la table InnoDB, avec la commande ALTER TABLE ... TYPE=INNODB, ou de crèer une table vide InnoDB avec les mêmes caractèristiques, et d’insèrer les lignes avec la commande INSERT INTO ... SELECT * FROM ....

Pour avoir un meilleur contrôle sur le processus d’insertion, c’est une bonne idèe de faire les insertions par portions de table :

```
INSERT INTO newtable SELECT * FROM oldtable
   WHERE votre_cle > borne1 AND votre_cle <= borne2;
```

Une fois que toutes les donnèes ont ètè insèrèes, vous pouvez renommer les tables.

Durant la conversion des grandes tables, vous devriez utiliser le buffer InnoDB en lui donnant une grande taille pour rèduire les accés disques. Ne lui donnez pas une taille supèrieure á 80% de votre mèmoire physique. Vous devriez aussi avoir de grand fichiers de log et un buffer de log important.

Assurez vous que vous avez toujours de l’espace pour les donnèes : les tables InnoDB prennent bien plus de place que les tables MyISAM. Si une commande ALTER TABLE rencontre un probléme d’espace disque, elle va s’annuler, et cela va prendre des heures si le disque est plein. Lors des insertions, InnoDB utilise un buffer d’insertion pour fusionner les index par groupe. Cela èconomise beaucoup d’accés disques. Lors d’une annulation, aucun mècanisme de ce type n’est utilisè, et l’annulation peut prendre jusqu’á 30 fois la durèe de l’insertion.

Dans le cas d’une annulation, si vous n’avez pas de donnèes importante dans votre base, il est mieux de terminer le processus et de dètruire les donnèes InnoDB et les fichiers de log, ainsi que le fichier de table ‘.frm’, et de recommencer, plutôt que d’attendre les millions d’accés disques.

7.5.4.2 Contraintes de clè ètrangéres

Depuis la version 3.23.43b, InnoDB supporte les contraîntes de clè ètrangéres. InnoDB est le premier gèstionnaire de tables MySQL qui permet de dèfinir des contraîntes de clè ètrangéres pour assurer la cohèrence de vos donnèes.

La syntaxe d'une clè ètrangére avec InnoDB :

```
[CONSTRAINT symbol] FOREIGN KEY (index_col_name, ...)
                    REFERENCES table_name (index_col_name, ...)
                    [ON DELETE CASCADE | ON DELETE SET NULL]
```

Les deux tables doivent être de type InnoDB et **il doit y avoir un index dans lesquelles les colonnes de clè ètrangére et de clè rèfèrencèe sont en premier**. InnoDB ne crèe pas automatiquement des index pour les clès ètrangéres et les clè rèfèrencèes : vous devez le faire vous même.

Les colonnes correspondantes dans la clè ètrangére doivent être de même type interne á InnDB, de façon á ce qu'elle puissent être comparèes sans conversion. La **taille et le signe des types entiers doivent être les mêmes**. La taille des chaînes n'a pas besoin d'être les mêmes.

Depuis la version 3.23.50, vous pouvez aussi associer la clause `ON DELETE CASCADE` ou `ON DELETE SET NULL` avec les contraintes de clè ètrangére.

Si `ON DELETE CASCADE` est spècifiè, et qu'une ligne de la table parente est effacèe, alors InnoDB va automatiquement effacer la lgine dans la table fille, dont la clè ètrangére est ègale á la clè rèfèrencèe dans la table parente. Si `ON DELETE SET NULL` est spècifiè, les lignes de la table fille sont automatiquement modifièe de façon á ce que les colonnes dans la clè ètrangére prennent automatiquement la valeur SQL de NULL.

Depuis la version 3.23.50, InnoDB ne vèrifie pas les contraintes de clè ètrangére pour ces clès ètrangéres, ou les clès qui font rèfèrence á la valeur NULL.

Depuis la version 3.23.50, InnoDB vous permet d'utiliser les guillemets amèricains (‘) autour des noms de tables et colonnes dans la clause `FOREIGN KEY ... REFERENCES ...` mais InnoDB ne prend pas encore en compte l'option `lower_case_table_names` que vous pouvez spècifier dans le fichier ‘`my.cnf`’.

Un exemple :

```
CREATE TABLE parent(id INT NOT NULL, PRIMARY KEY (id)) TYPE=INNODB;
CREATE TABLE child(id INT, parent_id INT, INDEX par_ind (parent_id),
            FOREIGN KEY (parent_id) REFERENCES parent(id)
            ON DELETE SET NULL
) TYPE=INNODB;
```

Si MySQL retourne une erreur numèro 1005 á la commande `CREATE TABLE`, et que l'erreur fait rèfèrence á au numèro 150, alors la crèation de table a èchouè car la clè ètrangére est mal formèe. De même, si une commande `ALTER TABLE` èchoue avec un numèro d'erreur de 150, cela signifie que la dèfinition de la clè ètrangére a ètè mal formèe dans la table.

Depuis la version 3.23.50, InnoDB vous permet d'ajouter une nouvelle clè ètrangére, comme ceci :

```
ALTER TABLE yourtablename
ADD [CONSTRAINT symbol] FOREIGN KEY (...) REFERENCES anothertablename(...)
```

N'oubliez pas de commencer par crèer vos index.

En InnoDB versions infèrieures á 3.23.50, `ALTER TABLE` ou `CREATE INDEX` ne doivent pas être utilisèe avec les tables qui ont une clè ètrangére, ou qui sont rèfèrencèes dans une clè ètrangére : toute commande `ALTER TABLE` supprime les contraintes de clè ètrangére dèfinie dans la table. Vous de devez pas utiliser la commande `ALTER TABLE` dans la table rèfèrencèe non plus, mais utilisez plutôt `DROP TABLE` et `CREATE TABLE` pour modifier le schèma. Lorsque MySQL fait un `ALTER TABLE` il peut utiliser en interne la commande `RENAME TABLE`, et cela va poser un probléme á la clè ètrangére á laquelle la table se rèfére. Une commande `CREATE INDEX` est traitèe par MySQL comme une commande `ALTER TABLE`, et les mêmes restrictions s'appliquent.

Lorsqu'il fait des vèrifications de clè ètrangéres, InnoDB pose des verrous de lignes sur les lignes filles et méres qu'il doit utiliser. InnoDB vèrifie les contraintes immèdiatement : la vèrification n'est pas reportèe á la validation de la transaction.

InnoDB vous permet d'effacer n'importe quelle table, même si cela casse la cohèrence d'une clè ètrangére, qui fait rèfèrence á la table. Lorsque vous effacez une table, les contraintes qui en faisaient parties sont aussi effacèes.

Si vous recrèez une table qui a ètè effacèe, elle doit avoir une dèfinition qui satisfont les contraintes de clè ètrangéres. Elle doit avoir les bons noms et types de colonnes, ainsi que les index nècessaires. Si ces conditions ne sont pas remplies, MySQL retourne une erreur numèro 1005, et fait rèfèrence au numèro de message 150.

Depuis la version 3.23.50, InnoDB retourne la dèfinition de contrainte de clè ètrangére lorsque vous appelez la commande

```
SHOW CREATE TABLE yourtablename
```

De plus, '`mysqldump`' produit la dèfinition correcte pour les tables dans l'export, et n'oublie pas les clès ètrangéres.

Vous pouvez aussi lister les clès ètrangéres pour une table `T` avec

```
SHOW TABLE STATUS FROM yourdatabasename LIKE 'T'
```

Les contraintes de clè sont listèes dans les commentaires de la table.

7.5.5 Ajouter et retirer des donnèes et des logs InnoDB

Depuis la verison 3.23.50 et 4.0.2, vous povuez spècifier le dernier fichier de donnèes InnoDB dans le fichier `autoextend`. Alternativement, vous pouvez augmenter votre espace de donnèes en spècifiant d'autres fichiers de donnèes. Pour cela, vous devez arrêter le serveur MYSQL, èditer le fichier '`my.cnf`' pour y ajouter un nouveau nom de fichier dans `innodb_data_file_path`, puis redèmarrer le serveur MySQL.

Actuellement, vous ne pouvez pas retirer de fichier de donnèes á InnoDB. Pour rèduire la taille de votre base de donnèes, vous devez utiliser l'utilitaire '`mysqldump`' pour exporter toutes vos donnèes de tables, crèer une nouvelle base, et rèimporter toutes vos donnèes dans cette nouvelle base.

Si vous voulez changer le nombre ou la taille de vos fichiers de log InnoDB, vous devez èteindre le serveur MYSQL et vous assurer qu'il s'est arrètè sans erreur. Puis, copiez les anciens fichiers de log dans une archive, car si vous rencontrez un probléme ultèrieurement, vous en aurez besoin pour restaurer votre base. Effacer les anciens fichiers de log du dossier

de logs, èditez le fichier '`my.cnf`', et redèmarrez le serveur MySQL. InnoDB vous indiquera au dèmarrage qu'il va crèer de nouveaux fichiers de log.

7.5.6 Sauver et restaurer une base InnoDB

Le secret de la gestion de bases de donnèes sereine rèside dans les sauvegardes règuliéres.

InnoDB Hot Backup est un outil de sauvegarde que vous pouvez utiliser pour faire des sauvegarde des bases de donnèes InnoDB, lorsqu'elles sont utilisèes. InnoDB Hot Backup ne vous impose pas de stopper le serveur, et ne pose aucun verrou ou ne perturbe votre utilisation normale des tables. InnoDB Hot Backup est un outils supplèmentaire, qui n,est pas libre, et qui n'est pas inclus dans la distribution standard de MySQL. Voyez le site web de InnoDB Hot Backup `http://www.innodb.com/hotbackup.html` pour plus de dètails et des captures d'ècran.

Si vous êtes capables d'arrèter votre serveur MySQL, alors pour faire une sauvegarde binaire de vos bases de donnèes, vous devez suivre les instructions suivantes :

- Arrêter votre serveur MySQL et assurez vous qu'il s'arrête sans erreur.
- Copiez tous les fichiers de donnèes dans une archive.
- Copiez tous les fichiers d'historique de InnoDB dans une archive.
- Copiez votre fichier de configuration '`my.cnf`' dans une archive.
- Copiez tous les fichiers '`.frm`' de vos tables InnoDB dans une archive.

En plus de rèaliser des copies des fichiers binaires tels que dècrit ci-dessus, il est recommandè de rèaliser des sauvegardes textuelles de vos tables avec '`mysqldump`'. La raison est qu'un fichier binaire peut être corrompu sans que vous ne vous en rendiez compte. Les tables sauvèes dans un fichier texte sont stockèes dans un format humainement lisible, bien plus simple qu'un fichier de base. Il est alors facile de rèpèrer une corruption des donnèes dans ce fichier, et comme le format de fichier est simple, il est facile de le corriger.

C'est une bonne idèe de prendre des copies de vos tables en même temps que vous faites une sauvegarde binaire. Vous devez èteindre tous les clients de votre base pour obtenir un bilan cohèrent de vos tables. Puis, vous pouvez faire les sauvegardes binaires, et vous aurez ainsi une sauvegarde cohèrente de votre base, en deux formats.

Pour être capable de restaurer des donnèes de votre base InnoDB jusqu'á prèsent, á partir des sauvegardes binaires ci-dessus, vous devez faire tourner le serveur MySQL en ayant activè le log gènèral et l'archivage des logs. Ici, par "log gènèral", nous considèrons le mècanisme gènèral de log de MySQL, qui est indèpendant des logs InnoDB.

Pour reconstruire une table aprés le crash d'un serveur MySQL, la seule chose que vous devez faire est de le redèmarrer. InnoDB va automatiquement vèrifier les fichiers d'historiques, et effectuer une mise á jour des donnèes. InnoDB va automatiquement annuler les transactions non validèes, qui ètaient en cours au moment du crash. Durant la restauration de la table, InnoDB va afficher des donnèes comme celles-ci :

```
~/mysqlm/sql > mysqld
InnoDB: Database was not shut down normally.
InnoDB: Starting recovery from log files...
InnoDB: Starting log scan based on checkpoint at
InnoDB: log sequence number 0 13674004
```

```
InnoDB: Doing recovery: scanned up to log sequence number 0 13739520
InnoDB: Doing recovery: scanned up to log sequence number 0 13805056
InnoDB: Doing recovery: scanned up to log sequence number 0 13870592
InnoDB: Doing recovery: scanned up to log sequence number 0 13936128
...
InnoDB: Doing recovery: scanned up to log sequence number 0 20555264
InnoDB: Doing recovery: scanned up to log sequence number 0 20620800
InnoDB: Doing recovery: scanned up to log sequence number 0 20664692
InnoDB: 1 uncommitted transaction(s) which must be rolled back
InnoDB: Starting rollback of uncommitted transactions
InnoDB: Rolling back trx no 16745
InnoDB: Rolling back of trx no 16745 completed
InnoDB: Rollback of uncommitted transactions completed
InnoDB: Starting an apply batch of log records to the database...
InnoDB: Apply batch completed
InnoDB: Started
mysqld: ready for connections
```

Si votre base se corrompt ou que votre disque est en panne, vous devez faire une restauration depuis une sauvegarde. En cas de corruption, vous devez trouver d'abord une sauvegarde qui n'est pas corrompue. A partir de cette sauvegarde, effectuer la restauration avec les fichiers d'historique gènèraux de MySQL, suivant les instructions du manuel MySQL.

7.5.6.1 Points de contrôle

InnoDB utilise un mècanisme de contrôle appelè points de contrôle flou. InnoDB va ècrire des blocs de donnèes modifièes depuis un buffer vers le disque par petits paquets : il n'y a pas besoin de tout ècrire en une seule fois, ce qui en gènèral, conduit á l'arrêt du traitement des autres instructions pour quelques instants.

Durant une restauration de base, InnoDB recherche un point de contrôle ècrit dans les fichiers de log. Il sait que toutes les modifications de la base placèes avant ce point de contrôle sont aussi prèsentes sur le disque de la base. Puis, InnoDB analyse le fichier de log, et applique les modifications qui ont eu lieu depuis le point de contrôle.

InnoDB ècrit dans les fichiers de log en mode circulaire. Toutes les modifications validèes qui font que le buffer de MySQL est diffèrent de la version sur le disque doivent être disponibles dans les fichiers de log, au cas oú InnoDB aurait besoin pour une restauration. Cela signifie que lorsque InnoDB commence á rèutiliser le fichier d'historique, il doit commencer par s'assurer que le disque a reçu les modifications qui sont dans le buffer. En d'autres termes, InnoDB doit placer un point de contrôle, et souvent, cela se traduit par l'ècriture de donnèes du buffer sur le disque.

Ceci explique pourquoi utiliser de grands fichiers de log peut èviter des accés disques pour les points de contrôle. Il est recommandè d'utiliser une taille de fichier d'historique aussi grande que le buffer, voire même plus grande. L'inconvènient des grands fichiers est que la restauration dure alors plus longtemps, puisqu'il y a plus de modifications á appliquer.

7.5.7 Transfèrer une base de donnèes InnoDB vers une autre machine

Les fichiers de donnèes et de logs de InnoDB sont compatibles en mode binaire sur toutes les plate-formes si le format des nombre á virgule flottante est le même. Vous pouvez dèplacer une base de donnèes InnoDB en copiant tous les fichiers concernès, que nous avons dèjá listès dans la section prècèdente traitant de la sauvegarde d'une base de donnèes. Si les formats des nombres á virgules flottantes sont diffèrents mais que vous n'avez pas utilisè les types de donnèes `FLOAT` ou `DOUBLE` dans vos tables alors la procèdure est lá même : copiez juste les fichiers concernès. Si les formats sont diffèrents et que vous utilisez de tels types de donnèes, vous devez utiliser '`mysqldump`' et '`mysqlimport`' pour transfèrer les tables.

Un bon moyen d'avoir de bonnes performances est de couper le mode auto-commit quand vous importez des donnèes dans votre base de donnèes, en supposant que votre tablespace posséde assez d'espace pour la grande partie d'annulations (rollback) que la grande transaction importèe gènére. Ne faites le commit qu'aprés avoir importè une table entiére, ou un segment de table.

7.5.8 Modéle transactionnel de InnoDB

Le modéle transactionnel d'InnoDB a pour but de combiner les avantages des bases de donnèes multi-version aux verrouillages traditionnels en deux phases. InnoDB fait un verrouillage de ligne, et exècute les requêtes par dèfaut avec des lectures cohèrentes non bloquante, de la même façon qu'Oracle. Les verrous InnoDB sont stockès de maniére efficace, pour que l'escalade de transaction ne soit pas nècessaire : typiquement, plusieurs utilisateurs sont autorisès á verrouiller toutes les lignes dans une base, ou un sous ensemble alèatoire de ligne, sans que InnoDB ne soit á court de mèmoire.

Avec InnoDB, toutes les activitès des utilisateurs sont placèes dans des transactions. Si le mode de validation automatique de MySQL est utilisè, chaque requête SQL reprèsente une transaction. MySQL dèmarre toujours une nouvelle connexion avec de mode activè.

Si le mode de validation automatique est dèsactivè, avec la commande `SET AUTOCOMMIT = 0`, alors nous considèrons que l'utilisateur a toujours une transaction ouverte. Si il utilise les commandes `COMMIT` ou `ROLLBACK`, cela va terminer la transaction courante, et en dèmarrer une nouvelle. Les deux commandes vont libèrer les verrous posès durant la transaction courante. Une validation `COMMIT` signifie que les modifications faites durant la transaction sont rendus permanents, et accessibles aux autres utilisateurs. Une annulation `ROLLBACK`, d'un autre cotè, annule toutes les modifications faites durant la transaction.

Si la connexion est en mode `AUTOCOMMIT = 1`, alors l'utilisateur peut rèaliser une transaction á requêtes multiples en commençant par la commande `BEGIN` et en terminant la transaction par `COMMIT` ou `ROLLBACK`.

En termes de niveau d'isolations de transaction SQL-1992, le mode par dèfaut d'InnoDB est `REPEATABLE READ`. Depuis la version 4.0.5, InnoDB offre 4 niveaux d'isolation de transaction diffèrents, tels que dècrits dans le standard SQL 1992. Vous pouvez choisir le niveau d'isolation par dèfaut du serveur `[mysqld]` dans le fichier de configuration '`my.cnf`' :

```
transaction-isolation = {READ-UNCOMMITTED | READ-COMMITTED
                         | REPEATABLE-READ | SERIALIZABLE}
```

Un utilisateur peut modifier le niveau d'isolation d'une transaction ou des nouvelles connexions avec la commande suivante :

```
SET [SESSION | GLOBAL] TRANSACTION ISOLATION LEVEL
                       {READ UNCOMMITTED | READ COMMITTED
                        | REPEATABLE READ | SERIALIZABLE}
```

Notez qu'il n'y a pas de tirets dans les noms de niveaux de la syntaxe SQL. Si vous spècifiez le mot rèservè `GLOBAL` dans la commande ci-dessus, cela va configurer les valeurs pour les nouvelles connexions, mais ne modifiera pas les valeurs des connexions dèjá ouvertes. Tout utilisateur est libre de changer le niveau d'isolation de sa session, même durant la transaction. Dans les versions infèrieure á 3.23.50, `SET TRANSACTION` n'avait aucun effet sur les tables InnoDB. Dans les versions infèrieures á 4.0.5, seuls `REPEATABLE READ` et `SERIALIZABLE` ètaient disponibles.

Vous pouvez connaître les valeurs de niveaux d'isolations global et courant avec la commande :

```
SELECT @@global.tx_isolation;
SELECT @@tx_isolation;
```

Durant le verrouillage de ligne, InnoDB utilise le verrouillage de clè suivante. Cela signifie qu'en plus des lignes de l'index, InnoDB verrouille aussi l'espace 'avant' un index, pour èviter les insertions d'un autre utilisateur. Un verrou de ligne suivante signifie le verrouillage de l'index et de l'intervalle entre cet index et le prècèdent. Un verrouillage d'espacement indique un verrouillage de l'espace entre deux index.

Voici une description plus dètaillèe des diffèrents niveaux d'isolation des transactions de InnoDB :

- `READ UNCOMMITTED`, qui est aussi appelè la 'lecture incohèrente' : les commandes `SELECT` non bloquantes sont exècutèes sans vèrifier la prèsence de versions antèrieures de la ligne : par consèquent, ces lectures ne sont pas cohèrentes avec ce niveau d'isolation. Sinon, ce niveau fonctionne exactement comme `READ COMMITTED`.
- `READ COMMITTED` Le niveau d'isolation qui se rapproche de celui d'Oracle. TOutes les commandes `SELECT ... FOR UPDATE` et `SELECT ... LOCK IN SHARE MODE` ne verrouille que les lignes d'index, et non pas les espaces qui les prècédent, et de cette maniére, ils autorisent les insertions entre deux lignes verrouillèes. Les commandes `UPDATE` et `DELETE` qui utilisent un index unique avec une condition de recherche, ne vont verrouiller que les index trouvès, et non pas les espaces prècèdent ces verrous. Mais avec les commandes `UPDATE` et `DELETE` en mode intervalle, InnoDB va poser des verrous de clè suivante, ou verrous d'espacement, et bloquer les insertions d'autres utilisateurs dans ces espaces. Ceci est nècessaire pour èviter la prèsence de lignes 'fantômes', qui sont nèfastes pour la rèplication. **Les lectures cohèrentes** fonctionnent comme sous Oracle : chaque lecture cohèrente, même durant une transaction, fixe et lit une version rècente de donnèes.
- `REPEATABLE READ` est le niveau d'isolation par dèfaut d'InnoDB. `SELECT ... FOR UPDATE`, `SELECT ... LOCK IN SHARE MODE`, `UPDATE`, et `DELETE` qui utilisent un index unique comme condition de recherche, ne vont verrouiller que la ligne d'index trouvèe, et non pas l'espace qui la prèpare. Sinon, ces opèrations vont employer le verrouillage de clès suivante, pour bloquer les insertions parasites. En **lectures cohèrentes** il y a une diffèrence importante avec le niveau d'isolation prècèdent : avec ce niveau, toutes

les lectures faites dans une même transaction sont faîtes avec la version ètablit lors de la premiére lecture. Cette convention signifie que si vous faites plusieurs lectures `SELECT` dans une même transaction, ces `SELECT` seront cohèrents entre eux.

- `SERIALIZABLE` Ce niveau ressemble au prècèdent, mais toutes les lectures `SELECT`s sont implicitement converties en `SELECT ... LOCK IN SHARE MODE`.

7.5.8.1 Lecture cohèrente

Une lecture cohèrente signifie que InnoDB utilise son systéme de multi-versionage pour prèsenter á une requête, une photo de la base á un moment donnè. La requête va alors voir les diffèrentses modifications apportèes par les transactions qui ont eu lieu avant cette date, et masquera les transactions ont eu lieu depuis, ou n'ont pas ètè archivèes. L'exception á cette régle est que la requête verra les modification apportèes la requête qui a èmis cette commande.

Si vous utilisez le niveau d'isolation `REPEATABLE READ`, alors les lectures cohèrentes dans une même transaction liront le même bilan. Vous pouvez obtenir un bilan plus rècent pour vos requêtes en archivant la requête courante, et en dèmarrant une autre.

Les lectures cohèrentes sont le mode par dèfaut de traitement des commandes `SELECT` par InnoDB avec les niveaux d'isolation `READ COMMITTED` et `REPEATABLE READ`. Une lecture cohèrente ne pose aucun verrou sur les tables auxquelles elle accéde, et par consèquent, les autres utilisateurs peuvent librement modifier ces tables en même temps qu'une lecture cohèrente est exècutèe.

7.5.8.2 Verrous de lecture

Une lecture cohèrente n'est pas toujours pratique, dans certaines circonstances. Supposons que vous voulez ajouter une ligne dans votre table `CHILD`, et vous assurer que l'enfant a dèjá un parent dans la table `PARENT`.

Supposez que vous utilisiez une lecture cohèrente, pour lire la table `PARENT`, et que vous dècouvrez le parent de l'enfant dans cette table. Pouvez vous ajouter tranquillement la ligne fille dans la table `CHILD`? Non, car il peut arriver que durant ce temps, un autre utilisateur a effacè la ligne parente dans la table `PARENT`, et vous n'en êtes pas conscient.

La solution est d'exècuter la commande `SELECT` en mode verrouillage, avec `LOCK IN SHARE MODE`.

```
SELECT * FROM PARENT WHERE NAME = 'Jones' LOCK IN SHARE MODE;
```

Effectuer une lecture en mode partagè signifie que vous allons lire les derniéres donnèes disponibles, et que nous allons poser un verrou sur les lignes que nous lisons. Si les derniéres donnèes appartiennent á une transaction non validèe d'un autre utilisateur, nous allons attendre que ce soit fait. Un verrou partagè èvite que les autres utilisateurs ne modifient ou n'effacent la ligne que nous lisons. Aprés que nous ayons obtenu le nom du parent, nous pouvons tranquillement ajouter le nom du fils dans la table `CHILD`, et valider notre transaction. Cet exemple montre comment implenter l'intègritè rèfèrentielle dans votre application.

Ajoutons un autre exemple : nous avons un champs compteur dans la table `CHILD_CODES` que nous utilisons pour assigner un identifiant unique á chaque enfant que nous ajoutons

dans la table CHILD. Evidemment, en utilisant une lecture cohèrente ou une lecture partagèe pour lire la valeur courante du compteur n'est pas une bonne idèe, car deux utilisateurs de la base peuvent simultanèment lire la même valeur de compteur, et nous allons obtenir une erreur de clè doublon lorsque nous ajouterons le second des deux fils.

Dans ce cas, il y a deux bonnes mèthodes pour implèmenter la lecture et l'incrèmentation du compteur : (1) modifiez le compteur d'une unitè, et lisez le aprés cela ou (2) lisez le compteur d'abord, avec un verrou en mode FOR UPDATE, puis incrèmentez le :

```
SELECT COUNTER_FIELD FROM CHILD_CODES FOR UPDATE;
UPDATE CHILD_CODES SET COUNTER_FIELD = COUNTER_FIELD + 1;
```

Une commande SELECT ... FOR UPDATE va lire les derniéres donnèes disponibles pour chaque ligne, et pose un verrou dessus en même tant qu'il lit. De cette façon, il pose le même verrou que la commande UPDATE.

7.5.8.3 Verrou de clè suivante : èviter le probléme des lignes fantômes

Avec le verrouillage de ligne, InnoDB utilise un algorithme appelè le verrouillage de la clè suivante. InnoDB fait un verrouillage de telle sorte que lorsqu'il fait une recherche ou un scan d'index, il pose des verrous partagès ou exclusifs sur les lignes d'index qu'il rencontre. Par consèquent, les verrous de lignes sont plus exactement appelès des verrous d'index.

Les verrous que InnoDB posent affectent aussi l'espace qui le sèpare de la ligne suivante. Si un utilisateur a un verrou partagè ou exclusif sur une ligne L dans un index, alors un autre utilisateur ne peut faire d'insertion immèdiatement avant la ligne L, dans l'ordre de l'index. Ce verrouillage est fait pour èviter le probléme de la ligne fantôme. Supposons que je veuille lire et verrouiller tous les enfants ayant un identifiant supèrieur á 100 dans la table CHILD, puis modifier certains champs des lignes ainsi identifièes :

```
SELECT * FROM CHILD WHERE ID > 100 FOR UPDATE;
```

Supposons qu'il y ait un index sur la table CHILD, sur la colonne ID. Notre requête va scanner l'index á partir de la premiére ligne oú ID est plus grand que 100. Maintenant, si le verrou posè sur les lignes d'index n'empêche pas l'utilisation des intervalles entre les lignes d'index, un nouvel enfant peut être insèrè dans la table durant la lecture. Et maintenant, si ma transaction exècute la commande :

```
SELECT * FROM CHILD WHERE ID > 100 FOR UPDATE;
```

je vais trouver un nouvel enfant dans le rèsultat de ma requête. Ceci va á l'encontre du principe d'isolation des transactions : une transaction doit être capable de s'exècuter sans que les lectures soient affectèes durant ce temps. Si vous considèrons un intervalle de connèes, alors la nouvelle ligne 'fantôme' va casser le principe d'isolation.

Lorsque InnoDB scanne un index, il va aussi verrouille l'espace aprés la derniére ligne de l'index. C'est ce qui est arrivè dans l'exemple ci-dessus : le verrou posè par InnoDB va èviter qu'une insertion n'intervienne dans la table oú ID serait plus grand que 100.

Vous pouvez utiliser le verrouillage de la clè suivant pour implèmenter des vèrifications d'unicitè dans votre application : si vous lisez des donnèes en mode partagè, et que vous ne voyez pas de duplicata de la ligne que vous allez insèrer, alors vous pouvez l'insèrer sans probléme, en sachant que le verrou de clè suivante va vous garantir que durant ce temps,

personne ne pourra insèrer de ligne, qui dèboucherait sur un duplicata de la votre. Le verrou de clè suivante permet de verrouiller aussi la non-existence de ligne dans votre table.

7.5.8.4 Les verrous posès par diffèrentes requêtes SQL avec InnoDB

- `SELECT ... FROM ...` : ceci est une lecture cohèrente, qui lit un bilan de la base, et ne pose aucun verrou.
- `SELECT ... FROM ... LOCK IN SHARE MODE` : pose un verrou partagè sur la prochaine clè sur tous les index que la lecture rencontre.
- `SELECT ... FROM ... FOR UPDATE` : pose un verrou exclusif sur la prochaine clè sur tous les index que la lecture rencontre.
- `INSERT INTO ... VALUES (...)` : pose un verrou exclusif sur la ligne insèrèe. Notez que ce verrou n'est pas un verrou de clè, et il n'empêche pas les autres utilisateurs d'insèrer des lignes. Si une erreur de clè double apparaît, un verrou sera posè partagè sera posè sur la ligne doublon.
- `INSERT INTO T SELECT ... FROM S WHERE ...` : pose un verrou exclusif sur chaque ligne insèrè dans `T`. Effectue la recherche sur `S` sous la forme d'une lecture cohèrente, mais pose un verrou partagè sur l'index de prochaine clè de `S` si MySQL a activè le log. `InnoDB` doit poser un verrou dans cette derniére situation, car en cas d'exècution des instructions dans une phase de restauration, toutes les requêtes doivent être exècutèes dans le même ordre.
- `CREATE TABLE ... SELECT ...` effectue une commande `SELECT` sous la forme d'une lecture cohèrente, ou avec des verrous partagès, comme prècèdemment.
- `REPLACE` est similaire á une insertion, si il n'y a pas de collision sur la clè unique. Sinon, une verrou exclusif sur l'index de prochaine clè est posè sur la ligne qui sera modifièe.
- `UPDATE ... SET ... WHERE ...` : pose un verrou exclusif sur l'index de prochaine clè, á chaque ligne que la recherche trouve.
- `DELETE FROM ... WHERE ...` : pose un verrou exclusif sur l'index de prochaine clè á chaque ligne que la recherche trouve.
- Si la contrainte de `FOREIGN KEY` est dèfinie sur une table, toute insertion, modification ou effacement qui requiert la vèrification de la contrainte va poser un verrou de ligne sur la ligne dont il doit vèrifier la contrainte. De plus, dans certains cas oú la contrainte èchoue, InnoDB pose ces verrous.
- `LOCK TABLES ...` : pose un verrou de table. L'implèmentation de la couche MySQL pose ce verrou. La dètection automatique des blocages de `InnoDB` ne peut dètecter les blocages lorsque de tels verrous sont posès. Voyez la section suivante. De plus, comme MySQL ne connaît pas le verrouillage de lignes, il est possible que vous posiez un verrou sur une table oú un autre utilisateur a dèjá posè un verrou. Mais cela ne pose pas de probléme quant á l'intègritè de la requête. Voir Section 7.5.13 [InnoDB restrictions], page 574.

7.5.8.5 Dètection des blocages et annulation

InnoDB dètecte automatiquement les blocages de transactions et annule une ou plusieurs transactions pour l'èviter. Depuis la version 4.0.5, InnoDB va essayer d'annuler les petites transactions. La taille de la transaction est dèterminèe par le nombre de lignes qu'elle a insèrè, modifiè ou effacè. Avant la version 4.0.5, InnoDB annulait toujours la transaction qui avait posè le dernier verrou avant le blocage, c'est á dire, un cycle dans le graphe des transactions.

InnoDB ne peut pas dètecter les blocages causès par la commande MySQL `LOCK TABLES`, ou si un verrou est posè par un autre gestionnaire de table que InnoDB. Vous devez rèsoudre ces situations avec l'option `innodb_lock_wait_timeout` du fichier de configuration '`my.cnf`'.

Lorsque InnoDB effectue une annulation de transaction, tous les verrous de cette transaction sont libèrès. Cependant, si une commande SQL est annulèe pour cause d'erreur, certains verrous de la transaction peuvent être conservès. Deci est dû au fait que InnoDB enregistre les verrous dans un format qui ne permet pas de savoir qui l'a posè.

7.5.8.6 Un exemple de lecture cohèrente avec InnoDB

Supposons que vous exècutez MySQL avec le niveau d'isolation des transactions de `REPEATABLE READ`. Lorsque vous demandez une lecture cohèrente avec une commande `SELECT` ordinaire, InnoDB va donner á votre transaction une date jalon, en fonction de laquelle votre requête va voir la base. Ainsi, si une transaction B efface une ligne aprés l'assignation de ce jalon, vous ne verrez pas cette ligne. De même pour une insertion ou une modification.

Vous pouvez faire avancer votre date jalon en validant votre transaction et en exècutant un autre `SELECT`.

Cela s'appelle le contrôle simultanè multi-versions.

```
                    User A                  User B

                SET AUTOCOMMIT=0;       SET AUTOCOMMIT=0;
time
|               SELECT * FROM t;
|               empty set
|                                        INSERT INTO t VALUES (1, 2);
|
v               SELECT * FROM t;
                empty set
                                        COMMIT;

                SELECT * FROM t;
                empty set;

                COMMIT;

                SELECT * FROM t;
```

```
--------------------
|    1    |    2    |
--------------------
```

De cette façon, A voir la ligne insèrèe par B uniquement lorsque B a validè son insertion, et que A a validè sa propre transaction, de façon á ce que la date jalon soit plus rècente que la validation de B.

Si vous voulez avoir une vision aussi "fraîche" que possible de votre base, vous devez utiliser le verrou en lecture :

```
SELECT * FROM t LOCK IN SHARE MODE;
```

7.5.8.7 Comment gèrer les blocages de verrous?

Les blocages de verrous sont un probléme classique des bases de donnèes transactionnelles, mais ils ne sont pas dangeureux, á moins qu'ils ne se rèpétent si souvent que vous ne puissiez pas exècuter tranquillement certaines transactions. Normalement, vous devriez ècrire vos applications de maniére á ce qu'elles soient prêtes á tenter á nouveau une transaction si la transaction est annulèe pour cause de blocage.

InnoDB utilise un verrouillage de lignes automatique. Vous pouvez obtenir des blocages sur une ligne, même si votre transactions ne fait que modifier ou insèrer une seule ligne. Cela est dû au fait que les opèrations ne sont pas rèellement 'atomiques' : elles posent automatiquement des verrous (èventuellement plusieurs) sur les lignes d'index de l'enregistrement concernè.

Vous pouvez gèrer ces blocages et rèduire leur nombre avec les trucs suivants :

- Utilisez la commande `SHOW INNODB STATUS` avec MySQL version supèrieure á 3.23.52 et 4.0.3, pour dèterminer la cause du dernier blocage. Cela peut vous aider á optimiser votre application pour les èviter.
- Soyez toujours prêts á tenter une nouvelle fois une transaction si elle èchoue á cause d'un blocage. Les verrous ne sont pas dangeureux. Essayez juste une autre fois.
- Validez souvent vos transactions. Les petites transactions sont moins sujettes aux blocages.
- Si vous utilisez des verrous en lectures avec `SELECT ... FOR UPDATE` ou `... LOCK IN SHARE MODE`, essayez d'utiliser un niveau d'isolation plus bas comme `READ COMMITTED`.
- Accèdez á vos tables et lignes dans un ordre fixè. Les transactions vont alors former des queues, et non pas des blocages.
- Ajoutez de bons index á vos tables. Vos requêtes devront scanner moins souvent les tables, et poseront donc moins de verrous. Utilisez `EXPLAIN SELECT` pour dèterminer si MySQL choisit les bons index pour vos requêtes.
- Limitez votre utilisation des verrous. Si vous pouvez vous permettre de faire retourner á une commande `SELECT` des donnèes un peu anciennes, n'ajoutez pas la clause `FOR UPDATE` ou `LOCK IN SHARE MODE`. Utiliser le niveau d'isolation `READ COMMITTED` est bon ici, car chaque lecture cohèrente dans la même transaction lira avec des donnèes aussi fraîches que possible á chaque fois.
- En dernier recours, vous pouvez forcer les verrous avec la commande : `LOCK TABLES t1 WRITE, t2 READ, ... ; [faire quelquechose avec les tables t1 et t2];`

`UNLOCK TABLES`. Les verrous de niveau de table forcent les transactions á se mettre en ligne, et les blocages sont èvitès. Notez que `LOCK TABLES` dèmarre implicitement une transaction, tout comme `BEGIN`, et `UNLOCK TABLES` termine une transaction avec un `COMMIT`.

- Une derniére solution est de crèer un sèmaphore auxiliaire sous la forme d'une table avec une seule ligne. Chaque transaction modifie cette table avant d'accèder aux autres tables. Dans ce cas, toutes les transactions se font en ordre sèquentiel. Notez que dans cette configuration, même l'algorithme InnoDB de dètection des blocages fonctionne, car le sèmaphore est un verrou de ligne. Avec les verrous de niveau de table de MySQL, nous devons nous rèsoudre á une mèthode de dèlai d'expiration pour rèsoudre un verrou.

7.5.8.8 Conseils sur l'amèlioration des performances InnoDB

1. Si l'outil Unix '`top`' ou si le '`Task Manager`' de Windows montre que l'utilisation du CPU, lors de la charge de travail, est infèrieure á 70%, votre calcul est probablement limitè par les disques. Vous faites peut être trop d'ècriture de transaction, ou le tampon de traitement ("buffer pool") est peut être trop petit. Augmenter la taille du tampon peut aider, mais il ne faut pas qu'il dèpasse 80% de la mèmoire physique.

2. Regrouper les modifications dans une seule transaction. InnoDB doit Wrap several modifications into one transaction. InnoDB must flush the log to disk at each transaction commit, if that transaction made modifications to the database. Since the rotation speed of a disk is typically at most 167 revolutions/second, that constrains the number of commits to the same 167/second if the disk does not fool the operating system.

3. If you can afford the loss of some latest committed transactions, you can set the '`my.cnf`' parameter `innodb_flush_log_at_trx_commit` to zero. InnoDB tries to flush the log anyway once in a second, though the flush is not guaranteed.

4. Make your log files big, even as big as the buffer pool. When InnoDB has written the log files full, it has to write the modified contents of the buffer pool to disk in a checkpoint. Small log files will cause many unnecessary disk writes. The drawback in big log files is that recovery time will be longer.

5. Also the log buffer should be quite big, say 8 MB.

6. (Relevant from 3.23.39 up.) In some versions of Linux and Unix, flushing files to disk with the Unix `fdatasync` and other similar methods is surprisingly slow. The default method InnoDB uses is the `fdatasync` function. If you are not satisfied with the database write performance, you may try setting `innodb_flush_method` in '`my.cnf`' to `O_DSYNC`, though O_DSYNC seems to be slower on most systems.

7. In importing data to InnoDB, make sure that MySQL does not have `autocommit=1` on. Then every insert requires a log flush to disk. Put before your plain SQL import file line

```
SET AUTOCOMMIT=0;
```

and after it

```
COMMIT;
```

If you use the '`mysqldump`' option `--opt`, you will get dump files which are fast to import also to an InnoDB table, even without wrapping them to the above `SET AUTOCOMMIT=0; ... COMMIT;` wrappers.

8. Beware of big rollbacks of mass inserts: InnoDB uses the insert buffer to save disk I/O in inserts, but in a corresponding rollback no such mechanism is used. A disk-bound rollback can take 30 times the time of the corresponding insert. Killing the database process will not help because the rollback will start again at the database startup. The only way to get rid of a runaway rollback is to increase the buffer pool so that the rollback becomes CPU-bound and runs fast, or delete the whole InnoDB database.

9. Beware also of other big disk-bound operations. Use `DROP TABLE` or `TRUNCATE` (from MySQL-4.0 up) to empty a table, not `DELETE FROM yourtable`.

10. Use the multi-line `INSERT` to reduce communication overhead between the client and the server if you need to insert many rows:

```
INSERT INTO yourtable VALUES (1, 2), (5, 5);
```

This tip is of course valid for inserts into any table type, not just InnoDB.

7.5.8.9 Le moniteur InnoDB

Depuis la version 3.23.41, InnoDB inclut le moniteur InnoDB (InnoDB Monitor) qui affiche des informations sur l'ètat interne des tables InnoDB. Lorsqu'il est activè, le moniteur InnoDB va faire afficher des donnèes au serveur MySQL '`mysqld`' sur la sortie standard (note : le client MySQL ne va rien afficher du tout), toutes les 15 secondes. Ces donnèes sont trés intèressantes pour optimiser les performances. Sous Windows, vous devez dèmarrer `mysqld-max` depuis la ligne de commande avec les options `--standalone --console` pour diriger le rèsultat de l'affichage á la console MS-DOS.

Il existe aussi `innodb_lock_monitor` qui affiche les mêmes informations que `innodb_monitor`, mais qui indique aussi les verrous posès par les transactions.

Les informations affichèes inclut des donnèes sur :

- les attentes de verrous par les transactions,
- Les attentes de sèmaphores pour les threads,
- les requêtes en attente d'accés aux fichiers,
- les statistiques sur les buffers, et
- les statistiques sur les purges et buffers d'insertion sur le thread principal InnoDB.

Vous pouvez dèmarrer le moniteur InnoDB avec la commande SQL suivante :

```
CREATE TABLE innodb_monitor(a int) type = innodb;
```

et le stopper avec

```
DROP TABLE innodb_monitor;
```

La syntaxe `CREATE TABLE` est simplement un moyen de passer une commande á une table InnoDB, via l'analyseur MySQL : la table crèèe n'est pas importante pour le moniteur InnoDB. Si vous interrompez le serveur lorsque le moniteur fonctionne, et que vous voulez redèmarrer le moniteur, vous devez alors effacer la table avant de pouvoir exècuter la même commande `CREATE TABLE` pour dèmarrer le moniteur. Cette syntaxe est susceptible de changer dans le futur.

Un exemple de rèsultat du moniteur InnoDB :

```
================================
010809 18:45:06 INNODB MONITOR OUTPUT
================================
--------------------------
LOCKS HELD BY TRANSACTIONS
--------------------------
LOCK INFO:
Number of locks in the record hash table 1294
LOCKS FOR TRANSACTION ID 0 579342744
TABLE LOCK table test/mytable trx id 0 582333343 lock_mode IX

RECORD LOCKS space id 0 page no 12758 n bits 104 table test/mytable index
PRIMARY trx id 0 582333343 lock_mode X
Record lock, heap no 2 PHYSICAL RECORD: n_fields 74; 1-byte offs FALSE;
info bits 0
 0: len 4; hex 0001a801; asc ;; 1: len 6; hex 000022b5b39f; asc ";;
 2: len 7; hex 000002001e03ec; asc ;; 3: len 4; hex 00000001;
...
-------------------------------------------
CURRENT SEMAPHORES RESERVED AND SEMAPHORE WAITS
-------------------------------------------
SYNC INFO:
Sorry, cannot give mutex list info in non-debug version!
Sorry, cannot give rw-lock list info in non-debug version!
-----------------------------------------------------
SYNC ARRAY INFO: reservation count 6041054, signal count 2913432
4a239430 waited for by thread 49627477 op. S-LOCK file NOT KNOWN line 0
Mut ex 0 sp 5530989 r 62038708 sys 2155035;
rws 0 8257574 8025336; rwx 0 1121090 1848344
-----------------------------------------------------
CURRENT PENDING FILE I/O'S
--------------------------
Pending normal aio reads:
Reserved slot, messages 40157658 4a4a40b8
Reserved slot, messages 40157658 4a477e28
...
Reserved slot, messages 40157658 4a4424a8
Reserved slot, messages 40157658 4a39ea38
Total of 36 reserved aio slots
Pending aio writes:
Total of 0 reserved aio slots
Pending insert buffer aio reads:
Total of 0 reserved aio slots
Pending log writes or reads:
Reserved slot, messages 40158c98 40157f98
Total of 1 reserved aio slots
Pending synchronous reads or writes:
```

```
Total of 0 reserved aio slots
-----------
BUFFER POOL
-----------
LRU list length 8034
Free list length 0
Flush list length 999
Buffer pool size in pages 8192
Pending reads 39
Pending writes: LRU 0, flush list 0, single page 0
Pages read 31383918, created 51310, written 2985115
----------------------------
END OF INNODB MONITOR OUTPUT
============================
010809 18:45:22 InnoDB starts purge
010809 18:45:22 InnoDB purged 0 pages
```

Quelques notes sur le rèsultat :

- Si la section LOCKS HELD BY TRANSACTIONS rapporte des attentes de verrous, alors votre application a des problémes de rètention de verrous. Le rapport peut vous aider á trouver les blocages de verrous, et les transactions bloquantes.
- La section SYNC INFO va vous indiquer les sèmaphores rèservès si vous compilez InnoDB avec la constante `UNIV_SYNC_DEBUG` dèfinie dans le fichier '`univ.i`'.
- La section SYNC ARRAY INFO rapporte les threads en attente d'un sèmaphore, et les statistiques d'attentes pour un mutex ou un verrou en lecture/ècriture. Les longues attentes peuvent être dues á des requêtes paralélles, ou des problémes avec la programmation des threads.
- La section CURRENT PENDING FILE I/O'S list les demandes d'accés aux fichiers. Un nombre important indique que le disque est un facteur limitant dans votre installation.
- La section BUFFER POOL vous donne des statistiques sur les pages qui ont ètè lues et ècrites. Vous pouvez calculer depuis ces nombres combien d'accés disques vos requêtes rèalisent actuellement.

7.5.9 Implèmentation du multi-versionnage

Comme InnoDB est une base de donnes multi-versionnèe, elle doit conserver des informations des vieilles versions des lignes dans l'espace de tables. Cette information est stockèe dans la structure de donnèes que nous appelons un segment d'annulation, d'aprés la structure similaire d'Oracle.

En interne, InnoDB ajoute deux champs á chaque ligne stockèe dans la base. Un champ de 6 octets note l'identifiant de la derniére transaction qui a insèrè ou modifier la ligne. De plus, un effacement est traitè en interne comme une modification, oú un bit spècial dans la ligne sert á marquer la ligne comme effacèe. Chaque ligne contient aussi une colonne de 7 octets appelè un pointeur d'annulation. Ce pointeur fait rèfèrence á une ligne dans un

fichier d'annulation qui contient les informations nècessaires pour reconstruire le contenu de la ligne qui a ètè modifièe.

InnoDB utilise les informations dans le segment d'annulation pour effectuer les opèrations d'annulation nècessaires dans une annulation de transaction. Il utilise aussi ces informations pour reconstruire les anciennes versions d'une ligne lors d'une lecture cohèrente.

Le log d'annulation dans le segment d'annulation est divisè entre les insertions et les modifications. Le log d'insertion n'est nècessaire que lors des annulations de transactions, et peut être vidè dés que la requête est validèe. Le log de modification est utilisè lors des lectures cohèrentes, et les informations y sont supprimèes une fois que toutes les transactions qui font une lecture cohèrente sont terminèes.

Vous devez vous rappeler que valider vos transactions règuliérement, même ces transactions qui ne font que des lectures cohèrentes. Sinon, InnoDB ne peut pas vider le log de modification, et le segment d'annulation va grossir ènormèment.

La taille physique d'une ligne du log d'annulation dans le segment d'annulation est typiquement plus petite que la ligne correspondante insèrèe ou modifièe. Vous pouvez utiliser ces informations pour calculer l'espace nècessaire á vos segments d'annulation.

Dans nos schèmas de multi-versionnage, une ligne n'est pas physiquement supprimèe de la table immèdiatement lorsque vous l'effacez avec une requête SQL. Uniquement lorsque InnoDB va supprimer les lignes du log de modification, il vaut aussi supprimer physiquement la ligne, et les index. Cette opèration d'effacement est appelèe une purge, et elle est plutôt rapide, et aussi rapide que la requête SQL elle-même.

7.5.10 Structures de tables et d'index

MySQL enregistre la structure de table dans le fichier '`.frm`', du dossier de donnèes. Mais les tables InnoDB ont aussi leur propre entrèe dans les tables internes InnoDB. Lorsque MySQL efface une table ou une base, il efface le ou les fichiers '`.frm`' et aussi les lignes correspondantes dans les tables d'administration InnoDB. C'est la raison qui fait que vous ne pouvez pas dèplacer les tables InnoDB entre les bases simplement en dèplaçant le fichier '`.frm`', et pourquoi `DROP DATABASE` ne fonctionnait pas sous InnoDB en MySQL versions <= 3.23.43.

Chaque table InnoDB a un index spècial appelè un index en grappe, oú les donnèes des lignes sont enregistrèes. Si vous dèfinissez une clè primaire pour votre table `PRIMARY KEY`, alors l'index de la clè primaire de la table sera un index en grappe.

Si vous ne dèfinissez pas de clè primaire pour votre table, InnoDB va gènèrer un index en grappe, ou les lignes sont ordonnèes dans l'ordre des identifiants que InnoDB assigne aux lignes de la table. L'identifiant de ligne vaut 6 octets, et s'accroît au fur et á mesure que les lignes sont ajoutèes. Les lignes sont alors ordonnèes dans leur ordre d'insertion.

Accèder á une ligne via l'index en grappe est rapide, car la ligne de donnèes sera dans la même page que l'index. Dans de nombreuses bases, les donnèes sont traditionnellement stockèes dans un autre endroit. Si la table est grand, l'index en grappe èconomise de nombreux accés disques, comparativement aux solutions traditionnelles.

Les lignes des index qui ne sont pas en grappe (ce sont les index secondaires) dans InnoDB, contiennent la valeur de la clè primaire de la ligne. InnoDB utilise cette clè primaire pour

rechercher la valeur de la ligne dans l'index en grappe. Notez que si la clè primaire est longue, les index secondaires utiliseront plus de place.

7.5.10.1 Structure physique d'un index

Tous les index de InnoDB sont des B-trees oú les lignes d'index sont stockèes dans un noeud terminal de l'arbre. La taille par dèfaut d'une page d'index est de 16ko. Lorsque de nouvelles lignes sont insèrèes, InnoDB essaie de laisser 1 / 16 de la page de libre pour les prochaines insertions et modifications dans les lignes d'index

Si les lignes d'index sont insèrèes dans un ordre sèquentiel (croissant ou dècroissant), les pages d'index rèsultantes seront environs pleines á 15/16. Si les lignes sont insèrèes dans un ordre alèatoire, les pages seront pleines de 1/2 á 15/16. Si le taux de remplissage d'une page d'index tombe á 1/2, InnoDB va essayer de contracter l'arbre d'index pour libèrer la page.

7.5.10.2 Bufferisation des insertions

Une situation courante dans les applications de base de donnèes apparaît lorsque la clè primaire est un identifiant unique, et que les nouvelles lignes sont insèrèes dans un ordre ascendant. Par consèquent, les insertions dans l'index en grappe ne nècessitent pas de lectures alèatoires dans le disque.

D'un autre cotè, les index secondaires sont gènèralement non-uniques, et les insertions surviennent dans un ordre alèatoire. Cela causerait de nombreux accés disques alèatoire, si InnoDB de disposait pas d'un mècanisme spècial.

Si une ligne doit être insèrèe dans un index secondaire non unique, InnoDB vèrifie si la page d'index fait partie du buffer. Dans ce cas, InnoDB va faire directement l'insertion dans une structure de buffer destinèe á l'insertion. Le buffer d'insertion est conservè petit, pour qu'il reste dans le buffer gènèral, et les insertions sont faites trés vite.

Le buffer d'insertion est pèriodiquement fusionnè avec l'arbre d'index secondaires dans la base. Souvent, nous fusionnons plusieurs insertions dans la même page de l'arbre d'index, et donc, nous èconomisons des accés disques. Il a ètè mesurè que les insertions sont jusqu'á 15 fois plus rapides de cette façon.

7.5.10.3 Index hash adaptatifs

Si une base de donnèes est suffisamment petite pour tenir en mèmoire, alors le plus rapide pour faire des requêtes est d'utiliser les index hash. InnoDB a un mècanisme automatique pour surveiller les recherches utilisant les index d'une table, et si InnoDB remarque que la requête pourrait profiter d'un index hash, un tel index est automatiquement constituè.

Mais notez que les index hash sont toujours bâtis á partir d'un index B-tree existant. InnoDB peut bâtir un index hash sur un prèfixe de taille arbitraire de clè B-tree, suivant le modéle de recherche que InnoDB remarque dans l'index B-tree. Un index hash peut être partiel : il n'est pas obligatoire que tout l'index B-tree soit mis en cache dans le pool. InnoDB va bâtir des index hash á la demande pour les tables dont les index sont souvent sollicitès.

En un sens, grâce au mècanisme d'index hash adaptatif, InnoDB s'adapte tout seul á la mèmoire interne, et se rapproche des architectures de bases en mèmoire vive.

7.5.10.4 Structure physique d'une ligne

- Chaque ligne d'index de InnoDB contient un entête de 6 octets. L'entête est utilisè pour lier des lignes consècutives ensembles, et aussi pour le verrouillage de ligne.
- Les lignes dans un index en grappe contient des champs pour toutes les colonnes dèfinies par l'utilisateur. De plus, il y a un champ de 6 octets pour l'identification de transaction, et un champs de 7 octets pour le pointeur d'annulation.
- Si l'utilisateur n'a pas dèfini de clè primaire pour la table, chaque ligne de l'index en grappe contient aussi une colonne supplèmentaire de 6 octets, qui sert d'identification.
- Chaque ligne d'index secondaire contient aussi les champs dèfinis pour la clè de l'index en grappe.
- Une ligne contient aussi un pointeur pour chaque champs de la ligne. Si la taille totale des champs reprèsentent moins de 128 octets, alors le pointeur fait 1 octets, sinon 2.

7.5.10.5 Comment une colonne de type auto-increment fonctionne avec InnoDB

Aprés le dèmarrage de la base, lorsqu'un utilise faire un premier insert dans la table `T` oú une colonne de type auto-increment a ètè dèfinie, et que l'utilisateur ne fournit pas de valeur pour la colonne, alors InnoDB exècute la requête `SELECT MAX(auto-inc-column) FROM T`, et assigne la valeur incrèmentèe de 1 dans la colonne, et dans le compteur d'auto-incrèmente de la table. Nous disons alors que le compteur de la table `T` a ètè initialisè.

InnoDB suit la même procèdure dans l'initialisation du compteur d'une nouvelle table.

Notez que si l'utilisateur spècifie la valeur de 0 dans l'insertion, InnoDB le traite comme une valeur non dèfinie.

Une fois que le compteur de table a ètè initialisè, si l'utilisateur insére une colonne dans laquelle il spècifie explicitement une valeur, et que la valeur est plus grande que la valeur du compteur, alors le compteur prend cette valeur spècifiée. Si l'utilisateur ne fournit pas de valeur, alors InnoDB incrèmente le compteur de 1, et assigne la nouvelle valeur á la colonne.

Le mècanisme auto-incrèment, lorsque vous assignez une valeur au compteur, ignore le verrouillage et la gestion de transaction. Par consèquent, vous pouvez avoir des trous dans le numèro de sèquence, si vous annulez une transaction qui a modifiè la valeur du compteur.

Le comportement des colonnes auto-incrèment n'est pas dèfinie si un utilisateur fourni un nombre nègatif pour la colonne, ou si la valeur est plus grande que le plus grand des entiers que peut stocker la colonne.

7.5.11 Gestion de l'espace fichiers et des entrèes/sorties disque

7.5.11.1 Accés disques

Lors des accés disque, InnoDB utilise un accés asynchrone. Sous Windows NT, il utilise le mode natif asynchrone, fourni par le systéme d'exploitation. Sous Unix, InnoDB utilise un mode asynchrone simulè, intègrè á InnoDB: InnoDB crèe un nombre de threads d'accés disques pour prendre en charge les opèrations disques, comme les lectures. Dans les prochaines versions, nous allons ajouter le support des accés disques simulès á Windows NT et les accés natifs á Unix.

Sous Windows NT, InnoDB utilise des accés non bufferisès. Cela signifie que les pages disques que InnoDB lit ou ècrit ne sont pas mis en cache par le systéme d'exploitation. Cela èconomise de la mèmoire.

Depuis la version 3.23.41, InnoDB utilise une nouvelle technique de flush de fichier, appelèe doublewrite. Elle apporte de la sècuritè lors de la restauration aprés crash du systéme d'exploitation, ou un probléme èlectrique, et amèliore les performances sous Unix, pour plusieurs distributions, en rèduisant le besoin de synchronisation.

Doublewrite (Double ècriture, en français) signifie que InnoDB, avant d'ècrire les pages dans le fichier de donnèes, les ècrits dans une zone continue d'espace, appelèe un buffer de double ècriture. Une fois que cette ècriture est faite, que le buffer de double ècriture a ètè vidè, InnoDB ècrit les pages á leur destination finale. Si le systéme d'exploitation crashe entre temps, InnoDB va trouver une bonne copie des donnèes dans le buffer, lors de la restauration.

Depuis la version 3.23.41, vous pouvez aussi utiliser une partition brute de disque comme fichier de donnèes, même si cela n'a pas ètè testè pour le moment. Lorsque vous crèe un nouveau fichier de donnèes, vous devez ajouter le mot clè `newraw` immèdiatement aprés la taille du fichier de donnèes, dans `innodb_data_file_path`. La partition doit être supèrieure ou ègale á cette taille. Notez que 1M pour InnoDB est 1024 x 1024 octets, alors que les spècifications de disques utilisent souvent la convention de 1 Mo vaut 1000 000 octets (un million).

```
innodb_data_file_path=hdd1:5Gnewraw;hdd2:2Gnewraw
```

Lorsque vous dèmarrez la base á nouveau, vous **devez** changer le mot clè á `raw`. Sinon, InnoDB va ècrire sur votre partition!

```
innodb_data_file_path=hdd1:5Graw;hdd2:2Graw
```

En utilisant un disque brut, vous pouvez rèaliser des opèrations non bufferisèe sous Unix.

InnoDB dispose de deux heuristiques de lectures : lecture sèquentielle et lecture alèatoire. En mode sèquentiel, InnoDB remarque que l'accés á un segment de la table est sèquentiel. Il va alors anticiper, et faire des lectures de plusieurs pages dans la base. En mode alèatoire, InnoDB note qu'une partie de la table est lue dans son entier. Il va alors anticiper, et faire automatiquement les derniéres lectures.

7.5.11.2 Gestion de l'espace fichier

Les fichiers de donnèes que vous dèfinissez dans le fichier de configuration forment l'espace de donnèes InnoDB. Les fichiers sont simplement concatènès pour former un espace de donnèes, et il n'y a pas de parallèlisme utilisè. Actuellement, vous ne pouvez pas spècifier

oú l'espace est allouè pour vos tables, hormis en utilisant la mèthode suivante : pour chaque nouvel espace de donnèes crèè, InnoDB va allouer de l'espace en partant depuis la fin.

L'espace de tables est constituè de pages de taille 16 ko. Les pages sont groupèes en ensembles de 64 pages consècutives. Les 'fichiers' dans un espace de tables sont appelès des segments en InnoDB. Le nom du segment d'annulation est un peu trompeur, car il contient en fait de nombreux segments dans l'espace de table.

Pour chaque index de InnoDB, nous allons crèer deux segments : un pour les noeuds non terminaux du B tree, et un autre pour les noeuds terminaux. L'idèe ici est d'amèliorer la sèquence des noeuds terminaux, qui contiennent les donnèes.

Lorsqu'un segment croit dans un espace de table, InnoDB alloue les 32 premiéres pages spècifiquement pour un segment. Aprés cela, InnoDB commence á allouer des extensions au segment. InnoDB peut ajouter aux grands segments jusqu'á 4 extension en même temps, pour amèliorer la sèquence de donnèes.

Certaines pages dans l'espace de table contiennent des cartes des autres pages, et donc, quelques extensions dans un espace de table InnoDB ne pourront pas être allouèes en tant que segment, mais comme pages individuelles.

Lorsque vous exècutez une commande `SHOW TABLE STATUS FROM ... LIKE ...` pour demander quel est l'espace libre dans la table, InnoDB va rapporter les extensions qui sont encore totalement libre. InnoDB se rèserve toujours quelques extensions pour la gestion interne. Ces extensions rèservèes ne sont pas inclues dans l'espace libre.

Lorsque vous effacez des donnes d'une table, InnoDB va contracter les index B-tree correspondant. Suivant les effacements qui vont libèrer des pages individuelles ou des extensions, de l'espace sera rendu aux autres tables. Effacer une table ou en effacer toutes les lignes va rendre obligatoirement de l'espace aux autres tables, mais souvenez vous que les lignes effacèes ne pourront être physiquement effacèe qu'aprés une opèration de purge, si aucune transaction n'en a plus besoin.

7.5.11.3 Dèfragmentation des tables

S'il y'a plusieurs insertions et suppressions dans les index d'une table, les index peuvent devenir fragmentès. Par fragmentation, nous voulons dire que l'ordre physique de la page d'index n'est pas proche de l'ordre alphabètique des enregistrements dans les pages, ou qu'il y'a plusieurs pages non-utilisèes dans le block de 64 pages qui ont ètè allouèes á l'index.

Une maniére d'accèlèrer les index est d'extraire pèriodiquement les donnèes de la table dans un fichier avec `mysqldump`, d'effacer la table puis de la recrèer.

Une autre maniére de dèfragmenter consiste á exècuter un `ALTER` sur le type de la table pour le changer en `MyISAM` puis de le repasser en `InnoDB`.

Notez qu'un table `MyISAM` doit pouvoir tenir sur un seul fichier de votre systéme d'exploitation.

Si les insertions dans un index sont toujours ascendantes et que les lignes supprimèes le sont á la fin du fichier, alors l'algorithme de gestion de l'espace fichiers de InnoDB garantit qu'aucune fragmentation n'aura lieu dans l'index.

7.5.12 Gestion des erreurs

La gestion des erreurs avec InnoDB n'est pas toujours la même que celle spècifièe par les standards ANSI SQL. En accord avec les standards ANSI, toute erreur survenant durant une commande SQL devrait provoquer l'annulation de celle-ci. InnoDB n'annule qu'une partie de la commande de temps en temps, ou la totalitè de la transaction. La liste suivante spècifie la gestion des erreurs de InnoDB.

- Si vous dèpassez l'espace d'un fichier dans l'espace des tables, vous obtiendrez l'erreur MySQL `'Table is full'` et InnoDB annulera la requête SQL.
- Un deadlock de transaction ou un dèpassement de temps dans un attente de verrouillage fait annuler toute la transaction á InnoDB.
- Une erreur de clef dupliquèe annule l'insertion de la ligne concernèe, même dans une requête du genre `INSERT INTO ... SELECT ...`. Cela fera probablement en sorte que la commande SQL sera annulèe si vous n'avez pas spècifiè l'option `IGNORE` dans votre requête.
- Une erreur 'row too long' annule la commande SQL.
- Les autres erreurs sont la plupart du temps dètectèes au niveau de la couche MySQL du code et annulent la commande SQL correspondante.

7.5.13 Restrictions sur les tables InnoDB

- **Attention** : **NE convertissez PAS** un systéme de tabel MyISAM en InnoDB! Cela n'est pas supportè. Si vous faîtes cela, MySQL ne va pas redèmarrer jusqu'á ce que vous restauriez vos donnèes avec une vieille sauvegarde, ou que vous regènèriez ces tables avec le script `mysql_install_db`.
- `SHOW TABLE STATUS` ne donne pas de statistiques prècises sur les tables InnoDB, hormis pour la taille physique rèservèe. Le nombre de lignes est seulement une approximation utilisèe en optimisation SQL.
- Si vous essayez de crèer un index unique sur un prèfixe d'une colonne, vous allez obtenir une erreur :

  ```
  CREATE TABLE T (A CHAR(20), B INT, UNIQUE (A(5))) TYPE = InnoDB;
  ```

  Si vous crèez un index non-unique sur un prèfixe de colonne, InnoDB va crèer un index pour toute la colonne.
- `INSERT DELAYED` n'est pas supportès par les tables InnoDB.
- La commande MySQL `LOCK TABLES` ne reconnait pas le verrouillage de ligne InnoDB rèalisè dans les commandes SQL achevèes : cela signifie que vous pouvez poser un verrou sur une table même si il existe une transaction qui a ètè posèe par un autre utilisateur. Par consèquent, votre opèration doit attendre que les autres tables soient libres, et elle peut aussi entrer en conflit avec une autre requête. De plus, un blocage de verrous est possible mais il ne met pas en danger l'intègritè des transactions, car le verrou de ligne posè par InnoDB se charge toujours de l'intègritè. Enfin, un verrou de table èvite aux autre transactions de poser un verrou de ligne (en conflit avec le mode de verrous) sur la table.
- Vous ne pouvez pas avoir une clè sur une colonne de type `BLOB` et `TEXT`.

- Une table ne peut pas recevoir plus de 1000 colonnes.
- `DELETE FROM TABLE` ne regènére pas la table, mais au lieu de cela, il efface les lignes une á une, ce qui est bien plus lent. Dans les prochaines versions, MySQL va pouvoir utiliser la commande `TRUNCATE` qui est trés rapide.
- La taille de page par dèfaut est de InnoDB est de 16 ko. En recompilant le code, vous pouvez lui changer la valeur de 8 ko á 64 ko. La taille maximale de ligne est lègérement infèrieure á la moitiè de la taille d'une page de base dans les versions infèrieure á 3.23.40 de InnoDB. Depuis la versions source 3.23.41 BLOB et TEXT sont autorisè á être plus petits que 4 Go, la taille totale de ligne doit aussi être de moins de 4 Go. InnoDB ne stocke par les fichiers dont le taille est infèrieure á 128 octets dans une page sèparèe. Aprés que InnoDB a modifiè la ligne en stockant les grandes colonnes dans une page sèparèe, le reste de la taille de la chaîen doit être plus petite que la moitièe de la page de base. La taille maximale de clè est de 7000 octets.
- Sur certains systémes d'exploitation, le fichier de donnèes est limitè á 2 Go. La taille combinèe des fichiers de log doit être infèrieure á 4 Go.
- La taille maximale d'un espace de tables est de 4 millions de pages. C'est aussi la taille maximale pour une table. La taille minimale pour une table est de 10 Mo.

7.5.14 Historique de l'èvolution InnoDB

Note : la traduction de cette section va commencer au moment du dèmarrage de la traduction gènèrale. Nous mettrons en version française les nouvelles entrèes. Les anciennes versions seront traduites en fonction du temps disponible.

7.5.14.1 MySQL/InnoDB-4.0.5, November 18, 2002

bullet InnoDB supporte dèsormais le niveau d'isolation de transaction transaction READ COMMITTED et READ UNCOMMITTED. READ COMMITTED èmule plus exactement Oracle et rend le portage des applications depuis Oracle bien plus faciles.

bullet La rèsolution des blocages de verrou est devenue sèlective : nous essayons de choisir les transactions qui ont modifiè le moins de ligne jusqu'á prèsent.

bullet La dèfinition des FOREIGN KEY prend en compte l'option `lower_case_table_names` de '`my.cnf`'.

bullet La ocmmande SHOW CREATE TABLE n'affiche pas le nom de la base dans une dèfinition de FOREIGN KEY si la table identifiè est dans la même base que celle de la table principale.

bullet InnoDB fait un texte de cohèrence pour la plupart des pages d'index avant de les ècrire dans le fichier de donnèes.

bullet Si vous utilisez `innodb_force_recovery` > 0, InnoDB essaie de passer outre les lignest et pages d'index corrompus, lors d'une commande SELECT * FROM dans une table. Cela aide lors de l'export.

bullet InnoDB utilise á nouveau les accés disques asynchrones sans buffer avec Windows 2000 et XP; seulement des accés disques avec simulation d'asynchronisme non bufferisè avec NT, 95/98/ME.

bullet Correction de bogue : l'estimateur de taille InnoDB exagèrait grandement la taille d'un intervalle court d'index, si les chemins jusqu'au point de fin dans l'arbre d'index ètait reliè á la racine. Cela conduisait á des scans de tables inutiles. La correction sera reportèe á la version 3.23.54.

bullet Correction d'un bogue prèsent dans les versions 3.23.52, 4.0.3, 4.0.4 : Le dèmarrage d'InnoDB pouvait prendre beaucoup de temps, et même crasher sur Windows 95/98/ME.

bullet Correction d'un bogue : le verrou AUTO-INC ètait conservè jusqu'á la fin de la transaction si il avait ètè donnè aprés une attete. Cela conduisait á des blocages inutiles.

bullet Correction d'un bogue : Si SHOW INNODB STATUS, innodb_monitor, ou innodb_lock_monitor devait afficher plusieurs centaines de transactions en un seul rapport, le rèsultat ètait tronquè, InnoDB s'arrêtait, affichait une erreur dans le log d'erreurs : log many waits for a mutex created at srv0srv.c, line 1621.

bullet Correction d'un bogue : SHOW INNODB STATUS sous Unix annoncait toujours une taille de lectures de fichier moyenne de 0 octets.

bullet Correction d'un bogue potentiel en 4.0.4: InnoDB fait dèsormais des clauses `ORDER BY ... DESC` comme `MyISAM`.

bullet Correction d'un bogue : DROP TABLE pour causer un crash ou un arrêt si il y avait des annulations simultanèes sur la même table. La correction sera appliquèe á la version 3.23 si cela apparait comme un vèritable probléme.

bullet Correction d'un bogue : ORDER BY pouvait èchouer, si vous n'aviez pas de clè primaire sur la table, mais que vous aviez dèfini plusieurs index, parmi lesquels au moins un index UNIQUE, et toutes les colonnes ètaient NOT NULL.

bullet Correction d'un bogue : une attente de verrou expirèe sur une connexion avec la commande `ON DELETE CASCADE` pouvait conduire á la corruption de la base.

bullet Correction d'un bogue : si une commande SELECT ètait faite avec une clè unique sur un index primaire, et que la recherche trouvait une ligne marquèe pour l'effacement, InnoDB pouvait retourner la ligne suivante.

bullet Bug identifiè : en 4.0.4, deux bogues ont ètè introduits avec les colonnes de type `AUTO_INCREMENT`. `REPLACE` peut faire que le compteur est corrompu, d'une unitè sous la vraie valeur. Un blocage de verrous ou une expiration de pose de verrou peut causer le même probléme. Cela sera corrigè en 4.0.6.

7.5.14.2 MySQL/InnoDB-3.23.53, October 9, 2002

bullet We again use unbuffered disk i/o to data files in Windows. Win XP and Win 2000 read performance seems to be very poor with normal i/o.

bullet Tuned range estimator so that index range scans are preferred over full index scans.

bullet Allow dropping and creating a table even if innodb_force_recovery is set. One can use this to drop a table which would cause a crash in rollback or purge, or if a failed table import causes a runaway rollback in recovery.

bullet Fixed a bogue present in 3.23.52, 4.0.3, 4.0.4: InnoDB startup could take very long or even crash on some Win 95/98/ME computers.

bullet Correction d'un bogue : fast shutdown (which is the default) sometimes was slowed down by purge and insert buffer merge.

bullet Correction d'un bogue : doing a big SELECT from a table where no rows were visible in a consistent read could cause a very long (> 600 seconds) semaphore wait in btr0cur.c line 310.

bullet Correction d'un bogue : the AUTO-INC lock was held to the end of the transaction if it was granted after a lock wait. This could cause unnecessary deadlocks.

bullet Correction d'un bogue : if you created a temporary table inside LOCK TABLES, and used that temporary table, that caused an assertion failure in ha_innobase.cc.

bullet Correction d'un bogue : if SHOW INNODB STATUS, innodb_monitor, or innodb_lock_monitor had to print several hundred transactions in one report, and the output became truncated, InnoDB would hang, printing to the error log many waits for a mutex created at srv0srv.c, line 1621.

bullet Correction d'un bogue : SHOW INNODB STATUS on Unix always reported average file read size as 0 bytes.

7.5.14.3 MySQL/InnoDB-4.0.4, October 2, 2002

bullet We again use unbuffered disk i/o in Windows. Win XP and Win 2000 read performance seems to be very poor with normal i/o.

bullet Increased the max key length of InnoDB tables from 500 to 1024 bytes.

bullet Increased the table comment field in SHOW TABLE STATUS so that up to 16000 characters of foreign key definitions can be printed there.

bullet The auto-increment counter is no longer incremented if an insert of a row immediately fails in an error.

bullet Allow dropping and creating a table even if innodb_force_recovery is set. One can use this to drop a table which would cause a crash in rollback or purge, or if a failed table import causes a runaway rollback in recovery.

bullet Correction d'un bogue : Using ORDER BY primarykey DESC in 4.0.3 causes an assertion failure in btr0pcur.c, line 203.

bullet Correction d'un bogue : fast shutdown (which is the default) sometimes was slowed down by purge and insert buffer merge.

bullet Correction d'un bogue : doing a big SELECT from a table where no rows were visible in a consistent read could cause a very long (> 600 seconds) semaphore wait in btr0cur.c line 310.

bullet Correction d'un bogue : if the MySQL query cache was used, it did not get invalidated by a modification done by ON DELETE CASCADE or ...SET NULL.

bullet Correction d'un bogue : if you created a temporary table inside LOCK TABLES, and used that temporary table, that caused an assertion failure in ha_innodb.cc.

bullet Correction d'un bogue : if you set innodb_flush_log_at_trx_commit to 1, SHOW VARIABLES would show its value as 16 million.

7.5.14.4 MySQL/InnoDB-4.0.3, August 28, 2002

bullet Removed unnecessary deadlocks when inserts have to wait for a locking read, update, or delete to release its next-key lock.

bullet The MySQL HANDLER SQL commands now work also for InnoDB type tables. InnoDB does the HANDLER reads always as consistent reads. HANDLER is a direct access path to read individual indexes of tables. In some cases HANDLER can be used as a substitute of server-side cursors.

bullet Fixed a bogue in 4.0.2: even a simple insert could crash the AIX version.

bullet Correction d'un bogue : if you used in a table name characters whose code is > 127, in DROP TABLE InnoDB could assert on line 155 of pars0sym.c.

bullet Compilation from source now provides a working version both on HP-UX-11 and HP-UX-10.20. The source of 4.0.2 worked only on 11, and the source of 3.23.52 only on 10.20.

bullet Correction d'un bogue : if compiled on 64-bit Solaris, InnoDB produced a bus error at startup.

7.5.14.5 MySQL/InnoDB-3.23.52, August 16, 2002

bullet The feature set of 3.23 will be frozen from this version on. New features will go the the 4.0 branch, and only bogue fixes will be made to the 3.23 branch.

bullet Many CPU-bound join queries now run faster. On Windows also many other CPU-bound queries run faster.

bullet A new SQL command SHOW INNODB STATUS returns the output of the InnoDB Monitor to the client. The InnoDB Monitor now prints detailed info on the latest detected deadlock.

bullet InnoDB made the SQL query optimizer to avoid too much index-only range scans and choose full table scans instead. This is now fixed.

bullet "BEGIN" and "COMMIT" are now added in the binlog around transactions. The MySQL replication now respects transaction borders: a user will no longer see half transactions in replication slaves.

bullet A replication slave now prints in crash recovery the last master binlog position it was able to recover to.

bullet A new setting innodb_flush_log_at_trx_commit=2 makes InnoDB to write the log to the operating system file cache at each commit. This is almost as fast as the setting innodb_flush_log_at_trx_commit=0, and the setting 2 also has the nice feature that in a crash where the operating system does not crash, no committed transaction is lost. If the operating system crashes or there is a power outage, then the setting 2 is no safer than the setting 0.

bullet Added checksum fields to log blocks.

bullet SET FOREIGN_KEY_CHECKS=0 helps in importing tables in an arbitrary order which does not respect the foreign key rules.

bullet SET UNIQUE_CHECKS=0 speeds up table imports into InnoDB if you have UNIQUE constraints on secondary indexes.

bullet SHOW TABLE STATUS now lists also possible ON DELETE CASCADE or ON DELETE SET NULL in the comment field of the table.

bullet When CHECK TABLE is run on any InnoDB type table, it now checks also the adaptive hash index for all tables.

bullet If you defined ON DELETE CASCADE or SET NULL and updated the referenced key in the parent row, InnoDB deleted or updated the child row. This is now changed to conform to SQL-92: you get the error 'Cannot delete parent row'.

bullet Improved the auto-increment algorithm: now the first insert or SHOW TABLE STATUS initializes the auto-inc counter for the table. This removes almost all surprising deadlocks caused by SHOW TABLE STATUS.

bullet Aligned some buffers used in reading and writing to data files. This allows using unbuffered raw devices as data files in Linux.

bullet Correction d'un bogue : If you updated the primary key of a table so that only the case of characters changed, that could cause assertion failures, mostly in page0page.ic line 515.

bullet Correction d'un bogue : If you delete or update a row referenced in a foreign key constraint and the foreign key check has to wait for a lock, then the check may report an erroneous result. This affects also the ON DELETE... operation.

bullet Correction d'un bogue : A deadlock or a lock wait timeout error in InnoDB causes InnoDB to roll back the whole transaction, but MySQL could still write the earlier SQL statements to the binlog, even though InnoDB rolled them back. This could, for example, cause replicated databases to get out-of-sync.

bullet Correction d'un bogue : If the database happened to crash in the middle of a commit, then the recovery might leak tablespace pages.

bullet Correction d'un bogue : If you specified a non-latin1 character set in my.cnf, then, in contrary to what is stated in the manual, in a foreign key constraint a string type column had to have the same length specification in the referencing table and the referenced table.

bullet Correction d'un bogue : DROP TABLE or DROP DATABASE could fail if there simultaneously was a CREATE TABLE running.

bullet Correction d'un bogue : If you configured the buffer pool bigger than 2 GB in a 32-bit computer, InnoDB would assert in buf0buf.ic line 214.

bullet Correction d'un bogue : on 64-bit computers updating rows which contained the SQL NULL in some column could cause the undo log and the ordinary log to become corrupt.

bullet Correction d'un bogue : innodb_log_monitor caused a hang if it suppressed lock prints for a page.

bullet Correction d'un bogue : in the HP-UX-10.20 version mutexes would leak and cause race conditions and crashes in any part of InnoDB code.

bullet Correction d'un bogue : if you ran in the AUTOCOMMIT mode, executed a SELECT, and immediately after that a RENAME TABLE, then RENAME would fail and MySQL would complain about error 1192.

bullet Correction d'un bogue : if compiled on 64-bit Solaris, InnoDB produced a bus error at startup.

7.5.14.6 MySQL/InnoDB-4.0.2, July 10, 2002

bullet InnoDB is essentially the same as InnoDB-3.23.51.

bullet If no innodb_data_file_path is specified, InnoDB at the database creation now creates a 10 MB auto-extending data file ibdata1 to the datadir of MySQL. In 4.0.1 the file was 64 MB and not auto-extending.

7.5.14.7 MySQL/InnoDB-3.23.51, June 12, 2002

bullet Correction d'un bogue : a join could result in a seg fault in copying of a BLOB or TEXT column if some of the BLOB or TEXT columns in the table contained SQL NULL values.

bullet Correction d'un bogue : if you added self-referential foreign key constraints with ON DELETE CASCADE to tables and a row deletion caused InnoDB to attempt the deletion of the same row twice because of a cascading delete, then you got an assertion failure.

bullet Correction d'un bogue : if you use MySQL 'user level locks' and close a connection, then InnoDB may assert in ha_innobase.cc, line 302.

7.5.14.8 MySQL/InnoDB-3.23.50, April 23, 2002

bullet InnoDB now supports an auto-extending last data file. You do not need to preallocate the whole data file at the database startup.

bullet Made several changes to facilitate the use of the InnoDB Hot Backup tool. It is a separate non-free tool you can use to take online backups of your database without shutting down the server or setting any locks.

bullet If you want to run the InnoDB Hot Backup tool on an auto-extending data file you have to upgrade it to version ibbackup-0.35.

bullet The log scan phase in crash recovery will now run much faster.

bullet Starting from this server version, the hot backup tool truncates unused ends in the backup InnoDB data files.

bullet To allow the hot backup tool to work, on Windows we no longer use unbuffered i/o or native async i/o; instead we use the same simulated async i/o as on Unix.

bullet You can now define the ON DELETE CASCADE or ON DELETE SET NULL clause on foreign keys.

bullet FOREIGN KEY constraints now survive ALTER TABLE and CREATE INDEX.

bullet We suppress the FOREIGN KEY check if any of the column values in the foreign key or referenced key to be checked is the SQL NULL. This is compatible with Oracle, for example.

bullet SHOW CREATE TABLE now lists also foreign key constraints. Also mysqldump no longer forgets about foreign keys in table definitions.

bullet You can now add a new foreign key constraint with ALTER TABLE ... ADD CONSTRAINT FOREIGN KEY (...) REFERENCES ... (...).

bullet FOREIGN KEY definitions now allow backquotes around table and column names.

bullet MySQL command SET TRANSACTION ISOLATION LEVEL ... has now the following effect on InnoDB tables: if a transaction is defined as SERIALIZABLE then InnoDB conceptually adds LOCK IN SHARE MODE to all consistent reads. If a transaction is defined to have any other isolation level, then InnoDB obeys its default locking strategy which is REPEATABLE READ.

bullet SHOW TABLE STATUS no longer sets an x-lock at the end of an auto-increment index if the auto-increment counter has already been initialized. This removes in almost all cases the surprising deadlocks caused by SHOW TABLE STATUS.

bullet Correction d'un bogue : in a CREATE TABLE statement the string 'foreign' followed by a non-space character confused the FOREIGN KEY parser and caused table creation to fail with errno 150.

7.5.14.9 MySQL/InnoDB-3.23.49, February 17, 2002

bullet Correction d'un bogue : if you called DROP DATABASE for a database on which there simultaneously were running queries, the MySQL server could crash or hang. Crashes fixed, but a full fix has to wait some changes in the MySQL layer of code.

bullet Correction d'un bogue : on Windows one had to put the database name in lower case for DROP DATABASE to work. Fixed in 3.23.49: case no longer matters on Windows. On Unix the database name remains case-sensitive.

bullet Correction d'un bogue : if one defined a non-latin1 character set as the default character set, then definition of foreign key constraints could fail in an assertion failure in dict0crea.c, reporting an internal error 17.

7.5.14.10 MySQL/InnoDB-3.23.48, February 9, 2002

bullet Tuned the SQL optimizer to favor more often index searches over table scans.

bullet Fixed a performance problem when several large SELECT queries are run concurrently on a multiprocessor Linux computer. Large CPU-bound SELECT queries will now also generally run faster on all platforms.

bullet If MySQL binlogging is used, InnoDB now prints after crash recovery the latest MySQL binlog file name and the position in that file (= byte offset) InnoDB was able to recover to. This is useful, for example, when resynchronizing a master and a slave database in replication.

bullet Added better error messages to help in installation problems.

bullet One can now recover also MySQL temporary tables which have become orphaned inside the InnoDB tablespace.

bullet InnoDB now prevents a FOREIGN KEY declaration where the signedness is not the same in the referencing and referenced integer columns.

bullet Correction d'un bogue : calling SHOW CREATE TABLE or SHOW TABLE STATUS could cause memory corruption and make mysqld to crash. Especially at risk was mysqldump, because it calls frequently SHOW CREATE TABLE.

bullet Correction d'un bogue : if on Unix you did an ALTER TABLE to an InnoDB table and simultaneously did queries to it, mysqld could crash with an assertion failure in row0row.c, line 474.

bullet Correction d'un bogue : if inserts to several tables containing an auto-inc column were wrapped inside one LOCK TABLES, InnoDB asserted in lock0lock.c.

bullet In 3.23.47 we allowed several NULLS in a UNIQUE secondary index. But CHECK TABLE was not relaxed: it reports the table as corrupt. CHECK TABLE no longer complains in this situation.

bullet Correction d'un bogue : on Sparc and other high-endian processors SHOW VARIABLES showed innodb_flush_log_at_trx_commit and other boolean-valued startup parameters always OFF even if they were switched on.

bullet Correction d'un bogue : if you ran mysqld-max-nt as a service on Windows NT/2000, the service shutdown did not always wait long enough for the InnoDB shutdown to finish.

7.5.14.11 MySQL/InnoDB-3.23.47, December 28, 2001

bullet Recovery happens now faster, especially in a lightly loaded system, because background checkpointing has been made more frequent.

bullet InnoDB allows now several similar key values in a UNIQUE secondary index if those values contain SQL NULLs. Thus the convention is now the same as in MyISAM tables.

bullet InnoDB gives a better row count estimate for a table which contains BLOBs.

bullet In a FOREIGN KEY constraint InnoDB is now case-insensitive to column names, and in Windows also to table names.

bullet InnoDB allows a FOREIGN KEY column of CHAR type to refer to a column of VARCHAR type, and vice versa. MySQL silently changes the type of some columns between CHAR and VARCHAR, and these silent changes do not hinder FOREIGN KEY declaration any more.

bullet Recovery has been made more resilient to corruption of log files.

bullet Unnecessary statistics calculation has been removed from queries which generate a temporary table. Some ORDER BY and DISTINCT queries will now run much faster.

bullet MySQL now knows that the table scan of an InnoDB table is done through the primary key. This will save a sort in some ORDER BY queries.

bullet The maximum key length of InnoDB tables is again restricted to 500 bytes. The MySQL interpreter is not able to handle longer keys.

bullet The default value of innodb_lock_wait_timeout was changed from infinite to 50 seconds, the default value of innodb_file_io_threads from 9 to 4.

7.5.14.12 MySQL/InnoDB-4.0.1, December 23, 2001

bullet InnoDB is the same as in 3.23.47.

bullet In 4.0.0 the MySQL interpreter did not know the syntax LOCK IN SHARE MODE. This has been fixed.

bullet In 4.0.0 multi-table delete did not work for transactional tables. This has been fixed.

7.5.14.13 MySQL/InnoDB-3.23.46, November 30, 2001

bullet This is the same as 3.23.45.

7.5.14.14 MySQL/InnoDB-3.23.45, November 23, 2001

bullet This is a boguefix release.

bullet In versions 3.23.42-.44 when creating a table on Windows you have to use lower case letters in the database name to be able to access the table. Fixed in 3.23.45.

bullet InnoDB now flushes stdout and stderr every 10 seconds: if these are redirected to files, the file contents can be better viewed with an editor.

bullet Fixed an assertion failure in .44, in trx0trx.c, line 178 when you drop a table which has the .frm file but does not exist inside InnoDB.

bullet Fixed a bogue in the insert buffer. The insert buffer tree could get into an inconsistent state, causing a crash, and also crashing the recovery. This bogue could appear especially in large table imports or alterations.

bullet Fixed a bogue in recovery: InnoDB could go into an infinite loop constantly printing a warning message that it cannot find free blocks from the buffer pool.

bullet Correction d'un bogue : when you created a temporary table of the InnoDB type, and then used ALTER TABLE to it, the MySQL server could crash.

bullet Prevented creation of MySQL system tables 'mysql.user', 'mysql.host', or 'mysql.db', in the InnoDB type.

bullet Fixed a bogue which can cause an assertion failure in 3.23.44 in srv0srv.c, line 1728.

7.5.14.15 MySQL/InnoDB-3.23.44, November 2, 2001

bullet You can define foreign key constraints on InnoDB tables. An example: FOREIGN KEY (col1) REFERENCES table2(col2).

bullet You can create > 4 GB data files in those file systems that allow it.

bullet Improved InnoDB monitors, including a new innodb_table_monitor which allows you to print the contents of the InnoDB internal data dictionary.

bullet DROP DATABASE will now work also for InnoDB tables.

bullet Accent characters in the default character set latin1 will be ordered according to the MySQL ordering.
 NOTE: if you are using latin1 and have inserted characters whose code is > 127 to an indexed CHAR column, you should run CHECK TABLE on your table when you upgrade to 3.23.43, and drop and reimport the table if CHECK TABLE reports an error!

bullet InnoDB will calculate better table cardinality estimates.

bullet Change in deadlock resolution: in .43 a deadlock rolls back only the SQL statement, in .44 it will roll back the whole transaction.

bullet Deadlock, lock wait timeout, and foreign key constraint violations (no parent row, child rows exist) now return native MySQL error codes 1213, 1205, 1216, 1217, respectively.

bullet A new my.cnf parameter innodb_thread_concurrency helps in performance tuning in high concurrency environments.

bullet A new my.cnf option innodb_force_recovery will help you in dumping tables from a corrupted database.

bullet A new my.cnf option innodb_fast_shutdown will speed up shutdown. Normally InnoDB does a full purge and an insert buffer merge at shutdown.

bullet Raised maximum key length to 7000 bytes from a previous limit of 500 bytes.

bullet Fixed a bogue in replication of auto-inc columns with multiline inserts.

bullet Fixed a bogue when the case of letters changes in an update of an indexed secondary column.

bullet Fixed a hang when there are > 24 data files.

bullet Fixed a crash when MAX(col) is selected from an empty table, and col is a not the first column in a multi-column index.

bullet Fixed a bogue in purge which could cause crashes.

7.5.14.16 MySQL/InnoDB-3.23.43, October 4, 2001

bullet This is essentially the same as InnoDB-3.23.42.

7.5.14.17 MySQL/InnoDB-3.23.42, September 9, 2001

bullet Fixed a bogue which corrupted the table if the primary key of a > 8000-byte row was updated.

bullet There are now 3 types of InnoDB Monitors: innodb_monitor, innodb_lock_monitor, and innodb_tablespace_monitor. innodb_monitor now prints also buffer pool hit rate and the total number of rows inserted, updated, deleted, read.

bullet Fixed a bogue in RENAME TABLE.

bullet Fixed a bogue in replication with an auto-increment column.

7.5.14.18 MySQL/InnoDB-3.23.41, August 13, 2001

bullet Support for < 4 GB rows. The previous limit was 8000 bytes.

bullet Use the doublewrite file flush method.

bullet Raw disk partitions supported as data files.

bullet InnoDB Monitor.

bullet Several hang bogues fixed and an ORDER BY bogue ('Sort aborted') fixed.

7.5.14.19 MySQL/InnoDB-3.23.40, July 16, 2001

bullet Only a few rare bogues fixed.

7.5.14.20 MySQL/InnoDB-3.23.39, June 13, 2001

bullet CHECK TABLE now works for InnoDB tables.

bullet A new my.cnf parameter innodb_unix_file_flush_method introduced. It can be used to tune disk write performance.

bullet An auto-increment column now gets new values past the transaction mechanism. This saves CPU time and eliminates transaction deadlocks in new value assignment.

bullet Several bogue fixes, most notably the rollback bogue in 3.23.38.

7.5.14.21 MySQL/InnoDB-3.23.38, May 12, 2001

bullet The new syntax SELECT ... LOCK IN SHARE MODE is introduced.

bullet InnoDB now calls fsync after every disk write and calculates a checksum for every database page it writes or reads, which will reveal disk defects.

bullet Several bogue fixes.

7.5.15 Informations de contact InnoDB

Informations de contact de Innobase Oy, producteur de InnoDB. Site web : `http://www.innodb.com/`. Courrier èlectronique : `Heikki.Tuuri@innodb.com`

```
phone: 358-9-6969 3250 (bureau) 358-40-5617367 (portable)
Innobase Oy Inc.
World Trade Center Helsinki
Aleksanterinkatu 17
P.O.Box 800
00101 Helsinki
Finland
```

7.6 Tables `BDB` ou `BerkeleyDB`

7.6.1 Vue d'ensemble des tables `BDB`

Le support des tables `BDB` est inclus par la distribution des sources de MySQL á partir de la version 3.23.34 et est activè dans le binaire MySQL-Max.

BerkeleyDB, disponible sur `http://www.sleepycat.com/` a fournit á MySQL un gestionnaire de tables transactionnelles. En utilisant les tables BerkeleyDB, vos tables ont plus de chances de survivre aux crashes, et vous avez accés á `COMMIT` et `ROLLBACK` avec les transactions. La distribution des sources de MySQL fournit une distribution patchèe de `BDB` pour lui permettre de fonctionner d'une façon plus souple avec MySQL. Vous pouvez utiliser une version non-patchèe de `BDB` avec MySQL.

Nous travaillons chez MySQL AB en coopèration ètroite avec Sleepycat pour maintenir une bonne qualitè d'interface MySQL/BDB.

Lorsqu'ils utiliseront les tables `BDB`, nous aiderons nos utilisateurs á trouver les problémes et á crèer des batteries de tests reproductibles pour tout probléme ayant trait aux tables `BDB`. De tels tests seront aussi envoyès á Sleepycat qui nous aidera aussi á trouver et rèsoudre les problémes. Vu qu'il s'agit d'une collaboration á deux niveaux, les problémes concernant les tables `BDB` prendront un peu plus de temps á être rèsolus en comparaison avec les autres gestionnaires de tables. Toutefois, ètant donè que le code de BerkeleyDB a lui-même ètè utilisè par plusieurs autres applications á part MySQL, nous n'envisageons pas de rencontrer de gros problémes avec. Voir Section 1.4.1 [Support], page 16.

7.6.2 Installation de BDB

Si vous avez tèlèchargè une version binaire de MySQL qui inclus le support de BerkeleyDB, vous n'avez qu'á suivre les instructions classiques. Voir Section 2.2.8 [Installing binary], page 82. Voir Section 4.7.5 [mysqld-max], page 304.

Pour compiler MySQL avec le support de Berkeley DB, tèlèchargez la version 3.23.34 ou plus de MySQL et configurez MySQL avec l'option --with-berkeley-db. Voir Section 2.3 [Installing source], page 84.

```
cd /chemin/vers/la/source/de/mysql-3.23.34
./configure --with-berkeley-db
```

Consultez le manuel fournit avec la distribution BDB pour des informations plus á jour.

Même si Berkeley DB a fait ses preuves, l'interface MySQL est encore considèrèe comme ètant de qualitè bêta. Nous sommes actuellement entrain de travailler á le rendre stable le plus rapidement possible.

7.6.3 Options de dèmarrage BDB

Si vous utilisez AUTOCOMMIT=0, vos changements dans les tables BDB ne seront pas effectuès tant que vous n'aurez pas utilisè COMMIT. Vous pouvez á la place utiliser ROLLBACK pour annuler les changements. Voir Section 6.7.1 [COMMIT], page 518.

Si vous utilisez AUTOCOMMIT=1 (par dèfaut), vos changements seront automatiquement pris en compte. Vous pouvez dèmarrer une transaction ètendue avec la commande SQL BEGIN WORK, aprés quoi vos changements ne seront pris en compte que si vous exècutez COMMIT (vous pouvez á tout moment annuler tous vos changements en exècutant ROLLBACK).

Les options suivantes de mysqld peuvent être utilisèes pour modifier le comportement des tables BDB :

Option	Description
--bdb-home=rèpertoire	Rèpertoire de base des tables BDB. Cela doit être le même rèpertoire que vous avez utilisès pour --datadir.
--bdb-lock-detect=#	Dètection des verrouillages Berkeley. (DEFAULT, OLDEST, RANDOM, ou YOUNGEST).
--bdb-logdir=rèpertoire	Rèpertoire des fichiers de log de Berkeley DB.
--bdb-no-sync	Ne pas vider les tampons synchroniquement.
--bdb-no-recover	Ne pas dèmarrer Berkeley DB en mode recouvrement.
--bdb-shared-data	Dèmarrer Berkeley DB en mode multi-processus (Ne pas utiliser DB_PRIVATE lors de l'initialisation de Berkeley DB)
--bdb-tmpdir=rèpertoire	Rèpertoire des fichiers temporaires de Berkeley DB.
--skip-bdb	Dèsactive l'utilisation des tables BDB.
-O bdb_max_lock=1000	Dèfinit le nombre maximal de verrous. Voir Section 4.5.6.4 [SHOW VARIABLES], page 273.

Si vous utilisez `--skip-bdb`, MySQL n'initialisera pas la librairie Berkeley DB et cela èconomisera beaucoup de mèmoire. Bien sûr, vous ne pouvez pas utiliser les table `BDB` si vous utilisez cette option. Si vous essayez de crèer une table `BDB`, MySQL crèera une table `MyISAM` á la place.

Normalement, vous devez dèmarrer `mysqld` sans `--bdb-no-recover` si vous avez l'intention d'utiliser des tables `BDB`. Cela peut cependant vous poser des problémes si vous essayez de dèmarrer `mysqld` alors que des fichiers de log `BDB` sont corrompus. Voir Section 2.4.2 [Starting server], page 102.

Vous pouvez spècifier le nombre maximal de verrous avec `bdb_max_lock` (10000 par dèfaut) que vous pouvez activer sur une table `BDB`. Vous devez l'augmenter si vous obtenez des erreurs du type `bdb: Lock table is out of available locks` ou `Got error 12 from ...` lorsque vous avez fait de longues transactions ou quand `mysqld` doit examiner beaucoup de lignes pour calculer la requête.

Vous voudrez peut-être aussi changer les options `binlog_cache_size` et `max_binlog_cache_size` si vous utilisez de grandes transactions multi-lignes. Voir Section 6.7.1 [`COMMIT`], page 518.

7.6.4 Caractèristiques des tables `BDB`

- Pour permettre les transactions annulabes, `BDB` gére un fichier de log. Pour maximiser les performances, vous devriez placer ces fichiers sur un autre disque que celui de votre base, en utilisant l'option `--bdb-logdir`.
- MySQL fait un point de contrôle á chaque fois qu'un nouveau fichier de log `BDB` est dèmarrè, et supprime les fichiers de logs anciens qui ne sont pas utiles. Si vous exècutez la commande `FLUSH LOGS`, vous placerez un nouveau point de contrôle pour les tables Berkeley DB.

  Pour la restauration aprés crash, vous devez utiliser les sauvegardes et le log binaire de MySQL. Voir Section 4.4.1 [Backup], page 242.

  Attention : si vous effacez les anciens fichiers de log qui sont en cours d'utilisation, `BDB` ne sera pas capable de faire la restauration et vous risquez de perdre des donnèes.
- MySQL requiert une clè `PRIMARY KEY` dans chaque table `BDB` pour être capable de faire rèfèrence aux lignes prècèdemment lues. Si vous n'en crèez pas, MySQL va gèrer une telle clè de maniére cachèe. La clè cachèe a une taille de 5 octets, et est incrèmentèe á chaque nouvelle insertion.
- Si toutes les colonnes auxquelles vous accèdez dans une table `BDB` font parties du même index dans la clè primaire, alors MySQL peut exècuter la requête sans avoir á lire la ligne elle-même. Dans une table`MyISAM`, ce qui prècéde n'est valable que si les colonnes font partie du même index.
- La clè `PRIMARY KEY` sera plus rapide que n'importe quelle autre clè, car la `PRIMARY KEY` est stockèe avec les donnèes. Comme les autres clès sont stockèes sous la forme donnèes `+ PRIMARY KEY`, il est important de garder une clè `PRIMARY KEY` aussi courte que possible pour èconomiser de l'espace disque, et amèliorer la vitesse.
- `LOCK TABLES` fonctionne avec les tables `BDB` sur les autres tables. Si vous n'utilisez pas le verrou `LOCK TABLE`, MySQL va poser un verrou interne multiple sur la table, pour s'assurer que la table est bien verrouillèe, si un autre thread tente de poser un verrou.

- Le verrouillage interne des tables BDB est fait au niveau page.
- SELECT COUNT(*) FROM table_name est trés lent, car les tables BDB ne maintiennent pas un compte de leur lignes dans la table.
- Scanner est plus lent qu'avec MyISAM car les tables BDB stockent les données dans un fichier B-tree et non pas dans un fichier sèparè.
- L'application doit toujours être prête á gèrer des cas oú une modification sur une table BDB peut être annulèe, ou une lecture abandonnèe pour cause de blocage de verrous.
- Les clès ne sont pas compressèes avec les clès prècèdentes, comme pour les tables ISAM et MyISAM. En d'autres termes, les informations de clès prennent un peu plus d'espace pour les tables BDB, comparativement aux tables MyISAM qui n'utilisent pas l'option PACK_KEYS=0.
- Il y a souvent des trous dans les tables BDB pour vous permettre d'insèrer de nouvelles lignes au milieu de l'arbre de donnèes. Cela rend les tables BDB un peu plus grandes que les tables MyISAM.
- L'optimiseur a besoin de connaître une approximation du nombre de lignes dans la table. MySQL rèsoud ce probléme en comptant les insertions et en conservant ce compte dans un segment sèparè pour chaque table BDB. Si vous ne faites pas souvent de DELETE ou ROLLBACK, ce nombre sera plutôt prècis pour l'optimiseur MySQL, mais comme MySQL ne stocke ce nombre qu'á la fermeture de la table, il peut être incorrecte si MySQL crashe. Cela ne doit pas être fatal si ce nombre n'est pas á 100% correct. Vous pouvez forcer la mise á jour de ce nombre avec la commande ANALYZE TABLE ou OPTIMIZE TABLE. Voir Section 4.5.2 [ANALYZE TABLE], page 264 . Voir Section 4.5.1 [OPTIMIZE TABLE], page 264.
- Si vous atteignez la capacitè maximale du disque avec la table BDB, vous allez obtenir une erreur (probablement l'erreur 28), et la transaction va s'annuler. C'est un comportement diffèrent des tables MyISAM et ISAM qui vont attendre que mysqld ait trouvè de l'espace disque avant de continuer.

7.6.5 Ce que nous devons corriger dans BDB dans un futur proche :

- Il est trés lent d'ouvrir de nombreuses tables BDB en même temps. Si vous utilisez des tables BDB, il ne faut pas avoir un cache de table trop grand (par exemple, > 256) et vous devriez utiliser l'option --no-auto-rehash avec le client mysql. Nous envisageons de corriger cela en partie en version 4.0.
- SHOW TABLE STATUS ne fourni pas encore beaucoup d'informations pour les tables BDB tables.
- Optimiser les performances.
- Ne pas utiliser les verrous de pages lorsque l'on scanne les tables.

7.6.6 Systémes d'exploitation supportès par BDB

Si, aprés avoir construit MySQL avec le support des tables BDB tables, obtenez l'erreur suivante dans le fichier de logs quand vous dèmarrez mysqld :

```
bdb: architecture lacks fast mutexes: applications cannot be threaded
```

```
Can't init dtabases
```

Cela signifie que les tables BDB ne sont pas supportèes par votre architecture. Dans ce cas, vous devez reconstruire MySQL sans le support des tables BDB.

Note : La liste suivante n'est pas compléte; nous la mettrons á jour au fur et á mesure que nous recevrons des informations á ce propos.

Actuellement, nous savons que le gestionnaire BDB fonctionne avec les systémes d'exploitation suivants :

- Linux 2.x intel
- Solaris SPARC
- Caldera (SCO) OpenServer
- Caldera (SCO) UnixWare 7.0.1

Il ne fonctionne pas sur les systémes d'exploitations suivants :

- Linux 2.x Alpha
- Max OS X

7.6.7 Restrictions avec les tables BDB

Voilá les restrictions que vous pouvez rencontrer en travaillant avec les tables BDB :

- Les tables BDB enregistrent dans le fichier '.db' le chemin vers le fichier tel qu'il ètait lorsqu'il a ètè crèe. (Cela est fait pour permettre de detecter les verrous dans un environnement multi-utilisateurs qui supporte les liens symboliques).

  Cela fait que les tables BDB ne peuvent être changèes de rèpertoire !
- Lors de la sauvegarde de tables BDB, vous devez utiliser mysqldump ou effectuer des sauvegardes de tout les fichiers nom_de_table.db et des fichiers de log BDB. Les fichiers de log de BDB sont les fichiers dans le rèpertoire de base des donnèes nommès log.XXXXXXXXXX (dix chiffres); Le gestionnaire des tables BDB stocke les transactions non-terminèes dans les fichiers de log et requiére la prèsence de ceux-ci lors du dèmarrage de mysqld.

7.6.8 Erreurs pouvant survenir lors de l'utilisation des tables BDB

- Si vous obtenez l'erreur suivante dans le fichier hostname.err log lors du dèmarrage de mysqld :

  ```
  bdb:  Ignoring log file: .../log.XXXXXXXXXX: unsupported log version #
  ```

  cela signifie que la nouvelle version de BDB ne supporte pas l'ancien format de log. Dans ce cas, vous devez effacer tous les logs BDB du dossier des donnèes (les fichiers dont le nom est au format log.XXXXXXXXXX) et redèmarrer mysqld. Nous vous recommandons aussi d'exècuter un mysqldump --opt de vos vieilles tables BDB, de les effacer, puis de restaurer les copies.
- Si vous n'êtes pas en mode auto-commit et que vous effacez une table qu'un autre thread utilise, vous obtiendrez le message d'erreur suivant dans le fichier d'erreurs de MySQL :

```
001119 23:43:56  bdb:  Missing log fileid entry
001119 23:43:56  bdb:  txn_abort: Log undo failed for LSN:
                      1 3644744: Invalid
```

Ce n'est pas une erreur trés grave mais nous ne vous recommandons pas d'effacer vos tables si vous n'êtes pas en mode auto-commit, tant que ce probléme n'est pas rèsolu (la solution n'est pas triviale).

8 Les interfaces pour MySQL

Ce chapitre dècrit les interfaces fournies par plusieurs langages pour MySQL, oú les obtenir, et comment les utiliser. L'interface C est la plus abordèe dans ce chapitre, puisqu'elle a ètè dèveloppèe par l'èquipe de MySQL, et qu'elle sert de base á toutes les autres interfaces.

8.1 API PHP pour MySQL

PHP est un langage côtè serveur, qui peut être utilisè pour crèer des pages web dynamiques. Il fournit un support d'accés á plusieurs systémes de bases de donnèes, dont MySQL fait partie. PHP peut être utilisè en tant que programme sèparè, ou être compilè en tant que module pour le serveur web Apache.

La distribution et documentation sont disponibles sur le site web de PHP (`http://www.php.net/`).

8.1.1 Problémes frèquents avec MySQL et PHP

- Error: "Maximum Execution Time Exceeded" C'est une limitation de PHP; Editez le fichier '`php.ini`' et changez le temps maximal d'exècution d'un script de 30 secondes á plus, selon vos besoins. C'est aussi une bonne idèe de doubler la quantitè de RAM allouèe par script en la changeant á 16MB.
- Error: "Fatal error: Call to unsupported or undefined function mysql_connect() in .." Cela signifie que votre version de PHP n'est pas compilèe avec le support de MySQL. Vous pouvez soit compiler un module dynamique MySQL et le charger dans PHP, ou bien recompiler PHP avec le support natif de MySQL. Ceci est dècrit en dètails dans le manuel PHP
- Error: "undefined reference to 'uncompress'" Cela signifie que la librairie cliente est compilèe avec support d'un protocole client/serveur compressè. La solution est d'ajouter `-lz` á la fin lors de la liaison avec `-lmysqlclient`.

8.2 API Perl pour MySQL

Cette section documente l'interface `DBI` de Perl. L'ancienne interface s'appelait `mysqlperl`. `DBI/DBD` est maintenant l'interface recommandèe avec Perl et donc `mysqlperl` est obsoléte et n'est pas documentèe ici.

8.2.1 `DBI` avec `DBD::mysql`

`DBI` est une interface gènèrique á plusieurs systémes de bases de donnèes. Cela signifie que vous pouvez ècrire un script qui fonctionne parfaitement avec plusieurs systémes diffèrents sans y apporter aucun changement. Vous avez besoin de dèfinir un pilote de base de donnèes (DataBase Driver : DBD) pour chaque systéme. Pour MySQL, ce pilote se nomme `DBD::mysql`.

Pour plus d'informations sur le module DBI de Perl5, visitez la page web de `DBI` et lisez la documentation :

```
http://dbi.perl.org/
```

Pour plus d'informations sur la programmation orientèe objets (OOP) comme la dèfinie Perl5, voyez la page OOP de Perl :

```
http://language.perl.com/info/documentation.html
```

Notez que si vous voulez utiliser les transactions avec Perl, vous avez besoin d'avoir au moins la version 1.2216 de `Msql-Mysql-modules`.

Les instructions pour l'installation du support Perl de MySQL sont donnèes dans Section 2.7 [Perl support], page 149.

8.2.2 L'interface `DBI`

Mèthodes portables

Mèthode	Description
`connect`	Etablit une connexion á un serveur de bases de donnèes.
`disconnect`	Dèconnecte du serveur de bases de donnèes.
`prepare`	Prèpare une commande SQL pour l'exècution.
`execute`	Exècute les requêtes prèparèes.
`do`	Prèpare et exècute une commande SQL.
`quote`	Echappe une chaîne ou une valeur `BLOB` avant insertion.
`fetchrow_array`	Rècupére la ligne suivante en tant que tableau de champs.
`fetchrow_arrayref`	Rècupére la ligne suivante comme une rèfèrence de tableau de champs.
`fetchrow_hashref`	Rècupére la ligne suivante en tant que rèfèrence pour un tableau associatif.
`fetchall_arrayref`	Rècupére toutes les donnèes comme un tableau de tableaux.
`finish`	Termine une requête et permet au systéme de libèrer les ressources.
`rows`	Retourne le nombre de lignes affectèes.
`data_sources`	Retourne un tableau de bases de donnèes disponibles sur le serveur local.
`ChopBlanks`	Contrôle la suppression des espaces avec `fetchrow_*`.
`NUM_OF_PARAMS`	Nombre de marqueurs dans la requête prèparèe.
`NULLABLE`	Les colonnes qui peuvent être `NULL`.
`trace`	Performe un traçage pour les corrections.

Mèthodes spècifiques á MySQL

Mèthode	Description
`insertid`	La deniére valeur `AUTO_INCREMENT`.
`is_blob`	Est-ce une valeur `BLOB` ?
`is_key`	Est-ce une clef ?
`is_num`	Est-ce une colonne numèrique ?
`is_pri_key`	Quelles colonnes sont des clefs primaires ?
`is_not_null`	Quelles colonnes NE peuvent PAS être `NULL`. Voir `NULLABLE`.

`length`	Taille maximale possible pour la colonne.
`max_length`	Taille maximale des colonnes actuellement prèsentes dans le rèsultat.
`NAME`	Noms des colonnes.
`NUM_OF_FIELDS`	Nombre de Number of fields returned.
`table`	Noms des tables dans le rèsultat.
`type`	Tout les types de colonnes.

Les mèthodes Perl sont dècrites dans les sections suivantes. Les variables utilisèe pour les valeurs de retour des mèthodes ont les significations suivantes :

`$dbh`	Gestionnaire de base de donnèes
`$sth`	Gestionnaire de requête
`$rc`	Code de retour (souvent un status)
`$rv`	Valeur de retour (souvent un comptage de lignes)

Mèthodes DBI portables

`connect($data_source, $username, $password)`

Utilise la mèthode `connect` pour crèe une connexion de base de donnèes á la source. La valeur doit commencer par `DBI:nom_pilote:`. Exemples d'utilisation de `connect` avec le pilote `DBD::mysql` :

```
$dbh = DBI->connect("DBI:mysql:$database", $user, $password);
$dbh = DBI->connect("DBI:mysql:$database:$hostname",
                    $user, $password);
$dbh = DBI->connect("DBI:mysql:$database:$hostname:$port",
                    $user, $password);
```

Si le nom d'utilisateur et/ou le mot de passe ne sont pas dèfinis, `DBI` utilises la valeur des variables d'environnements `DBI_USER` et `DBI_PASS`, respectivement. Si vous ne spècifiez pas un nom d'hôte, il est á `'localhost'` par dèfaut. Si vous ne spècifiez pas un numèro de port, il est mit par dèfaut au port par dèfaut de MySQL (3306).

Depuis la version 1.2009 de `Msql-Mysql-modules`, la valeur de `$data_source` accepte certains modificateurs :

`mysql_read_default_file=nom_du_fichier`

Lit le fichier '`filename`' en tant que fichier d'options. Pour plus d'informations sur les fichiers d'options, consulter Section 4.1.2 [Option files], page 198.

`mysql_read_default_group=nom_du_groupe`

Le groupe par dèfaut lors de la lecture d'un fichier d'options est normalement le groupe `[client]`. En spècifiant l'option `mysql_read_default_group`, le groupe par dèfaut devient le groupe `[nom_du_groupe]`.

`mysql_compression=1`

Utiliser la communication compressèe entre le client et le serveur (á partir de la version 3.22.3 de MySQL).

`mysql_socket=/path/to/socket`
: Spècifie le chemin vers la socket Unix qui est utilisèe pour se connecter au serveur. (MySQL version 3.21.15 ou plus).

Plusieurs modificateurs peuvent être donnès, ils doivent être sèparès par des point-virgules.

Par exemple, si vous voulez èviter d'avoir á ècrire le nom d'utilisateur et le mot de passe en dur dans un script `DBI`, vous pouvez le prendre du fichier utilisateur d'options '`~/.my.cnf`' en ècrivant votre appel á `connect` de la façon suivante :

```
$dbh = DBI->connect("DBI:mysql:$database"
                    . ";mysql_read_default_file=$ENV{HOME}/.my.cnf",
                    $user, $password);
```

Cet appel lira les options dèfinies pour le groupe `[client]` dans le fichier d'options. Si vous voulez le faire aussi pour le groupe `[perl]` par exemple, vous pouvez utiliser ce qui suit :

```
$dbh = DBI->connect("DBI:mysql:$database"
                    . ";mysql_read_default_file=$ENV{HOME}/.my.cnf"
                    . ";mysql_read_default_group=perl",
                    $user, $password);
```

`disconnect`
: La mèthode `disconnect` dèconnecte le gestionnaire de base de donnèes du serveur. On y fait appel avant de terminer le programme.

Exemple :

```
$rc = $dbh->disconnect;
```

`prepare($statement)`
: Prèpare une requête SQL pour l'exècution par le serveur de base de donnèes et retourne un gestionnaire de requête (`$sth`), que vous pouvez utiliser pour invoquer la mèthode `execute`. Le plus souvent, vous gèrez les requêtes `SELECT` (et ceux qui s'en rapprochent, comme `SHOW`, `DESCRIBE`, et `EXPLAIN`) á l'aide de `prepare` et `execute`.

Exemple :

```
$sth = $dbh->prepare($statement)
    or die "Impossible de prèparer $statement: $dbh->errstr\n";
```

`execute`
: La mèthode `execute` exècute une requête prèparèe. Pour les commandes qui ne se rapprochent pas de `SELECT`, `execute` retourne le nombre de lignes affectèes. Si aucune ligne ne l'est, `execute` retourne `"0E0"`, que Perl traite comme zèro mais considére comme vrai (true). Si une erreur survient, `execute` retourne `undef`. Pour les requêtes `SELECT`, `execute` ne fait que dèmarrer la requête SQL dans la base de donnèes; vous avez besoin d'utiliser une des mèthodes `fetch_*` dècrites ici pour rècupèrer les donnèes.

Exemple :

```
$rv = $sth->execute
          or die "ne peut exècuter la requête : $sth->errstr;
```

`do($statement)`

La mèthode `do` prèpare et exècute une requête SQL et retourne le nombre de lignes affectèes. Si aucune ne l'est, `do` retourne `"0E0"`, que Perl traite comme zèro mais considére comme vrai (true). Cette mèthode est gènèralement utilisèe pour les requêtes qui ne se rapprochent pas de `SELECT` qui ne peuvent être prèparèes á l'avance (á cause d'une limitation du pilote) ou qui n'ont pas besoin d'être exècutèes plus d'une fois (insertions, suppressions, etc.).

Exemple :

```
$rv = $dbh->do($statement)
        or die "Ne peut exècuter la commande $statement: $dbh- >errstr\n"
```

Gènèralement, la commande 'do' est plus rapide (et prèfèrable) á prepare/execute pour les requêtes ne contenant pas de paramétres.

`quote($string)`

La mèthode `quote` est utilisèe pour protèger les caractéres spèciaux contenus dans la chaîne et ajouter les guillemets externes requis.

Exemple :

```
$sql = $dbh->quote($string)
```

`fetchrow_array`

Cette mèthode rècupére la ligne de donnèes suivante et la retourne en tant que tableau de valeurs de champs.

Exemple :

```
while(@row = $sth->fetchrow_array) {
        print qw($row[0]\t$row[1]\t$row[2]\n);
}
```

`fetchrow_arrayref`

Cette mèthode rècupére la ligne de donnèes suivante et la retourne en tant que rèfèrence de tableau de valeurs de champs.

Exemple :

```
while($row_ref = $sth->fetchrow_arrayref) {
        print qw($row_ref->[0]\t$row_ref->[1]\t$row_ref->[2]\n);
}
```

`fetchrow_hashref`

Cette mèthode rècupére la ligne de donnèes et retourne une rèfèrence vers un tableau associatif contenant des paires nom du champ / valeur. Cette mèthode n'est pas aussi efficace que l'utilisation des rèfèrences sur tableaux comme expliquè plus haut.

Exemple :

```
while($hash_ref = $sth->fetchrow_hashref) {
        print qw($hash_ref->{firstname}\t$hash_ref->{lastname}\t\
                $hash_ref- > title}\n);
}
```

`fetchall_arrayref`
: Cette mèthode est utilisèe pour faire en sorte que toutes les donnèes (lignes) soient retournèes par la requête SQL. Elle retourne une rèfèrence á un tableau de rèfèrences á des tableaux pour chaque ligne. Vous pouvez accèder aux donnèes en utilisant des boucles profondes.

  Exemple :

```
my $table = $sth->fetchall_arrayref
                or die "$sth->errstr\n";
my($i, $j);
for $i ( 0 .. $#{$table} ) {
        for $j ( 0 .. $#{$table->[$i]} ) {
                print "$table->[$i][$j]\t";
        }
        print "\n";
}
```

`finish`
: Indique que plus aucune donnèe ne sera rècupèrèe du gestionnaire de requête. Vous appelez cette mèthode pour libèrer le gestionnaire de connexion et les ressources systéme lui ètant associèes.

  Exemple :

```
$rc = $sth->finish;
```

`rows`
: Retourne le nombre de lignes changèes (mises á jour, supprimèes, etc.) par la derniére commande. En l'utilise gènèralement aprés les requêtes autres que `SELECT` exècutèes avec la mèthode `execute`.

  Exemple :

```
$rv = $sth->rows;
```

`NULLABLE`
: Retourne une rèfèrence vers un tableau de valeurs qui indiquent si les colonnes peuvent contenir des valeurs `NULL`. Les valeurs possibles pour chaque èlèment sont 0 ou la chaîne vide si la colonne ne peut être `NULL`, 1 si elle peut, et 2 si l'ètat `NULL` de la colonne est inconnu.

  Exemple :

```
$null_possible = $sth->{NULLABLE};
```

`NUM_OF_FIELDS`
: Cet attribut indique le nombre de champs retournès par une requête `SELECT` ou `SHOW FIELDS`. Vous pouvez l'utiliser pour savoir si une requête a retournè un rèsultat : une valeur nulle indique que la requête n'est pas une sèlection, comme `INSERT`, `DELETE`, ou `UPDATE`.

  Exemple :

```
$nr_of_fields = $sth->{NUM_OF_FIELDS};
```

`data_sources($driver_name)`
: Cette mèthode retourne un tableau contenant les noms des bases de donnèes disponibles pour le serveur MySQL sur l'hôte `'localhost'`.

  Exemple :

```
@dbs = DBI->data_sources("mysql");
```

`ChopBlanks`

Cet attribut dètermine si les mèthodes `fetchrow_*` enlévent les caractéres blancs au dèbut et á la fin des valeurs retournèes.

Exemple :

```
$sth->{'ChopBlanks'} =1;
```

`trace($trace_level)`
`trace($trace_level, $trace_filename)`

La mèthode `trace` active ou dèsactive le traçage. Lorsqu'elle est invoquèe en tant que mèthode de la classe `DBI`, elle affecte le traçage á tout les gestionnaire. Lorsqu'elle est invoquèe comme un gestionnaire de base de donnèes ou de requête, elle n'affecte le traçage que pour le gestionnaire donnè (et tout ses descendants). Mettre `$trace_level` á 2 fournit des informations dètaillèes de traçage. Le mettre á 0 dèsactive le traçage. Les sorties de traçages vont dans la sortie d'erreurs standards par dèfaut. Si `$trace_filename` est spècifiè, le fichier est ouvert et le pointeur de fichier est placè á la fin de celui-ci et les sorties de **tout** les gestionnaires tracès sont ècrits dans ce fichier.

Exemple :

```
DBI->trace(2);                  # trace tout
DBI->trace(2,"/tmp/dbi.out"); # trace tout dans
                                # /tmp/dbi.out
$dth->trace(2);                 # trace ce gestionnaire de base de donnèes
$sth->trace(2);                 # trace ce gestionnaire de requête
```

Vous pouvez aussi activer le traçage `DBI` en dèfinissant la variable d'environnement `DBI_TRACE`. La dèfinir en tant que valeur numèrique revient á appeler `DBI->(value)`. La dèfinir en tant que chemin revient á appeler `DBI->(2,value)`.

Mèthodes spècifiques á MySQL

Les mèthodes montrèes ici sont spècifiques á MySQL et ne font pas partie du standard `DBI`. Beaucoup d'entre-elles sont aujourd'hui dèsapprouvèes : `is_blob`, `is_key`, `is_num`, `is_pri_key`, `is_not_null`, `length`, `max_length`, et `table`. Lorsque des alternatives utilisant le standard `DBI` existent, elles sont mentionnèes ici :

`insertid` Si vous utilisez la fonctionnalitè `AUTO_INCREMENT` de MySQL, la nouvelle valeur auto-incrèmentèe sera stockèe ici.

Exemple :

```
$new_id = $sth->{insertid};
```

Vous pouvez, en alternative, utiliser `$dbh->{'mysql_insertid'}`.

`is_blob` Retourne une rèfèrence vers un tableau de valeurs boolèennes; pour chaque èlèment du tableau, une valeur TRUE indique que la colonne en question est un `BLOB`.

Exemple :

```
$keys = $sth->{is_blob};
```

`is_key` Retourne une rèfèrence vers un tableau de valeurs boolèennes; pour chaque èlèment du tableau, une valeur TRUE indique que la colonne en question est une clef.

Exemple :

```
$keys = $sth->{is_key};
```

`is_num` Retourne une rèfèrence vers un tableau de valeurs boolèennes; pour chaque èlèment du tableau, une valeur TRUE indique que la colonne en question contient une valeur numèrique.

Exemple :

```
$nums = $sth->{is_num};
```

`is_pri_key`

Retourne une rèfèrence vers un tableau de valeurs boolèennes; pour chaque èlèment du tableau, une valeur TRUE indique que la colonne en question est une clef primaire.

Exemple :

```
$pri_keys = $sth->{is_pri_key};
```

`is_not_null`

Retourne une rèfèrence vers un tableau de valeurs boolèennes; pour chaque èlèment du tableau, une valeur FALSE indique que la colonne peut contenir `NULL` values.

Exemple :

```
$not_nulls = $sth->{is_not_null};
```

`is_not_null` est dèsapprouvè; il est prèfèrable d'utiliser l'attribut `NULLABLE` (dècrit plus haut), car c'est un standard DBI.

`length`
`max_length`

Chacune de ces mèthodes retourne une rèfèrence vers un tableau de tailles de colonnes. Le tableau `length` indique les tailles maximales de colonnes (comme dèclarès lors de la crèation de la table). Le tableau `max_length` indique la plus grande taille occupèe jusqu'á prèsent dans les colonnes de la table.

Exemple :

```
$lengths = $sth->{length};
$max_lengths = $sth->{max_length};
```

`NAME` Retourne une rèfèrence vers un tableau de noms de colonnes.

Exemple :

```
$names = $sth->{NAME};
```

`table` Retourne une rèfèrence vers un tableau de noms de tables.

Exemple :

```
$tables = $sth->{table};
```

`type` Retourne une rèfèrence vers un tableau de types de colonnes.

Exemple :

```
$types = $sth->{type};
```

8.2.3 Plus d'informations relatives á DBI/DBD

Vous pouvez utiliser la commande `perldoc` pour obtenir plus d'informations á propos de `DBI`.

```
perldoc DBI
perldoc DBI::FAQ
perldoc DBD::mysql
```

Vous pouvez aussi utiliser les outils `pod2man`, `pod2html`, etc., pour traduire en diffèrents formats.

Vous pouvez trouver les derniéres informations relatives á `DBI` á l'adresse suivante : `http://dbi.perl.org/`.

8.3 Support ODBC avec MySQL

MySQL fournit un support de ODBC via le programme `MyODBC`. Ce chapitre vous apprendra á installer `MyODBC`, et á l'utiliser. Vous trouverez aussi une liste de programmes connus pour leur utilisation de `MyODBC`.

8.3.1 Comment installer MyODBC

`MyODBC` 2.50 est un pilote 32-bit ODBC 2.50 avec un niveau de spècification 0 (avec le niveau 1 et 2 de proposès) pour connecter une application compatible ODBC á MySQL. `MyODBC` fonctionne sur Windows 9x/Me/NT/2000/XP et la plupart des plate-formes Unix. `MyODBC` 3.51 est une version amèliorèe avec les spècifications de niveau 1 de ODBC 3.5x (API noyau compléte + fonctionnalitès du niveau 2).

`MyODBC` est `Open Source`, et vous pouvez trouver la version la plus rècente sur `http://www.mysql.com/downloads/api-myodbc.html`. Notez que les version 2.50.x sont licencèes `LGPL` tandis que les versions 3.51.x sont licencèes `GPL`.

Si vous avez des problémes avec `MyODBC` et que votre programme fonctionne aussi avec OLEDB, essayez le pilote OLEDB.

Normalement, vous n'avez besoin d'installer `MyODBC` que sur les machines Windows. Vous avez besoin d'installer `MyODBC` sous Unix si vous avez un programme tel que ColdFusion qui fonctionne sur les machines Unix et utilise ODBC pour se connecter aux bases de donnèes.

Si vous voulez installer `MyODBC` sur un ordinateur Unix, vous aurez aussi besoin d'un gestionnaire `ODBC`. `MyODBC` est connu pour fonctionner avec la plupart des gestionnaires ODBC d'Unix. Voir Section 1.6.1 [Portals], page 23.

Pour installer `MyODBC` sur Windows, vous devez tèlècharger le fichier '`.zip`' de `MyODBC` appropriè, le dècompresser avec `WinZIP` ou un programme similaire et exècuter le fichier '`SETUP.EXE`'.

Sur Windows/NT/XP vous pouvez obtenir l'erreur suivante durant l'installation de `MyODBC` :

```
An error occurred while copying C:\WINDOWS\SYSTEM\MFC30.DLL. Restart
Windows and try installing again (before running any applications which
use ODBC)
```

Le probléme dans ce cas est qu'un autre programme utilise ODBC et du fait de l'architecture Windows, vous ne pouvez pas installer de nouveau pilote ODBC avec le programme d'installation de Microsoft ODBC. Dans la plupart des cas, vous pouvez continuer en clickant juste sur `Ignore` pour copier le reste des fichiers MyODBC et l'installation finale devrait fonctionner. Si ce n'est pas le cas, la solution est de redèmarrer votre machine en mode "safe mode" (faites le en appuyant sur F8 juste avant que votre machine ne dèmarre Windows), installez `MyODBC`, et redèmarrez en mode normal.

- Pour crèer une connexion á un ordinateur Unix depuis un ordinateur Windows, avec une application ODBC (une qui ne supporte pas MySQL nativement), vous devez installer `MyODBC` sur l'ordinateur Windows.
- L'utilisateur et la machine Windows doivent avoir les droits d'accés au serveur MySQL situè sur la machine Unix. vous pouvez configurer cela avec la commande `GRANT`. Voir Section 4.3.1 [`GRANT`], page 226.
- Vous devez crèer une entrèe DNS ODBC comme suit :
  - Ouvrez le panneau de configuration de Windows.
  - Double-clickez sur l'icône Sources de donnèes ODBC (32 bits).
  - Clickez sur le volet User DSN.
  - Clickez sur le bouton Add.
  - Sèlectionnez MySQL dans l'ècran Create New Data Source et clickez sur le bouton Terminer.
  - L'ècran de configuration par dèfaut du pilote MySQL est affichè. Voir Section 8.3.2 [ODBC administrator], page 600.
- Dèmarrez maintenant votre application et sèlectionnez le pilote ODBC avec les DSN que vous avez spècifiè dans l'administrateur ODBC.

Notez que d'autres options de configuration sont prèsentes dans l'ècran de MySQL (traçage, se connecter automatiquement, etc.), vous pouvez les essayer en cas de problémes.

8.3.2 Comment remplir les diffèrents champs dans le programme d'administrateur ODBC

Il y'a trois possibilitès pour indiquer le nom du serveur sous Windows95 :

- Utiliser l'adresse IP du serveur.
- Ajouter un fichier '`\windows\lmhosts`' avec les informations suivantes :

  ```
  ip nomhote
  ```

  Pour exemple :

  ```
  194.216.84.21 mon_nom_hote
  ```
- Configurer le PC pour qu'il utilise DNS.

Exemple de valeurs dans l'`installation ODBC` :

```
Windows DSN name:   test
Description:        Ceci est ma base de donnèes
MySql Database:     test
Server:             194.216.84.21
```

```
User:               monty
Password:           mon_pass
Port:
```

La valeur du champ `Windows DSN name` est n'importe quel nom qui est unique pour votre installation de ODBC sur Windows.

Vous n'avez pas besoin de spècifier des valeurs pour les champs `Server`, `User`, `Password`, et `Port` dans l'ècran d'installation d'ODBC. Toutefois, si vous le faites, ces valeurs seront celles utilisèes par dèfaut par la suite lorsque vous tenterez d'ètablir une connexion. Vous pourrez changez les options á ce moment lá.

Si le numèro du port n'est pas donnè, le port par dèfaut (3306) est utilisè.

Si vous spècifiez l'option `Read options from C:\my.cnf`, les groupes `client` et `odbc` seront lus á partir du fichier '`C:\my.cnf`'. Vous pouvez utiliser toutes les options utilisables par `mysql_options()`. Voir Section 8.4.3.38 [`mysql_options()`], page 641.

8.3.3 Paramétres de connexion de MyODBC

Il est possible de spècifier les paramétres suivants á `MyODBC` dans la section `[Servername]` du fichier '`ODBC.INI`' ou bien, avec l'argument `InConnectionString` dans l'appel de `SQLDriverConnect()`.

Paramétre	Valeur par dèfaut	Commentaire
user	ODBC (sous Windows)	Le nom d'utilisateur utilisè pour se connecter á MySQL.
server	localhost	Le nom d'hôte du serveur MySQL.
database		La base de donnèes par dèfaut.
option	0	Un entier avec lequel vous spècifier le mode de fonctionnement de `MyODBC`. Voir ci-dessous.
port	3306	Le port TCP/IP á utiliser si le `server` n'est pas `localhost`.
stmt		Une commande qui sera exècutèe au moment de la connexion á `MySQL`.
password		Le mot de passe pour le serveur `server`, et l'utilisateur `user`.
socket		La socket ou le pipe Windows á utiliser.

L'argument option est utilisè pour indiquer á `MyODBC` que le client n'est pas 100% compatible avec ODBC. Sous Windows, il est possible de configurer cette option en activant les options dans l'ècran de connexion, mais il est aussi possible de le configurer dans ce paramétre. Les options suivantes sont listèes dans l'ordre dans lequel elles apparaissent dans l'ècran `MyODBC` :

Bit	Description
1	Le client ne peut gèrer le fait que `MyODBC` retourne la taille rèelle de la colonne.
2	Le client ne peut gèrer que MySQL retourne le nombre de ligne affectè. Si cette option est activèe, alors MySQL retourne 'found rows' á la place. Il faut avoir MySQL 3.21.14 ou plus rècent pour profiter de cette fonctionnalitè.
4	Ecrit un fichier de dèbogage dans c:\myodbc.log. Cela revient au même que le code `MYSQL_DEBUG=d:t:O,c::\myodbc.log` dans le fichier '`AUTOEXEC.BAT`'
8	Ne limite pas les paquets pour les rèsultats et les paramétres.

16	Ne pas poser de questions, même si le pilote le veut.
32	Simule un pilote ODBC 1.0 dans certains contexte.
64	Ignore l'utilisation du nom de base de donnèes dans la syntaxe 'base.table.colonne'.
128	Force l'utilisation du gestionnaire de curseur ODBC (expèrimental).
256	Dèsactive l'utilisation de la lecture ètendue (expèrimental).
512	Compléte les champs CHAR jusqu'á contenance.
1024	SQLDescribeCol() retourne les noms de colonnes complets.
2048	Utilise le protocole de communication client/serveur
4096	Indique au serveur qu'il doit ignorer les espaces aprés les noms de fonction, et avant le `'('` (nècessaire pour PowerBuilder). Cela va faire de tous les noms de fonctions des mots clès.
8192	Connexion, avec les pipes nommès, á `mysqld` sur un machine NT.
16384	Change les colonnes LONGLONG en colonne INT (certaines applications ne peuvent pas gèrer les LONGLONG).
32768	Retourne 'user' comme Table_qualifier et Table_owner dans les tables SQLTables (expèrimental)
65536	Lit les paramétres des groupes `client` et `odbc` dans '`my.cnf`'
131072	Ajoute des vèrifications de sècuritè (ne devrait pas être nècessaire, mais...)

Si vous voulez combiner des options, vous devez additionner les options ci-dessus. Par exemple, activer l'option 12 (4+8), active le dèboggage sans limite de paquets.

La librairie par dèfaut '`MYODBC.DLL`' est compilèe pour des performances optimales. Si vous voulez dèboguer `MyODBC` (par exemple, pour activer le traçage), vous devriez utiliser '`MYODBCD.DLL`'. Pour installer ce ficher, copiez '`MYODBCD.DLL`' á la place de la librairie isntallè '`MYODBC.DLL`'.

8.3.4 Comment reporter les problémes avec ODBC

`MyODBC` a ètè testè avec Access, Admndemo.exe, C++-Builder, Borland Builder 4, Centura Team Developer (formerly Gupta SQL/Windows), ColdFusion (on Solaris and NT with svc pack 5), Crystal Reports, DataJunction, Delphi, ERwin, Excel, iHTML, FileMaker Pro, FoxPro, Notes 4.5/4.6, SBSS, Perl DBD-ODBC, Paradox, Powerbuilder, Powerdesigner 32 bit, VC++, et Visual Basic.

Si vous avez connaissance de n'importe quelle autre application qui utilise `MyODBC`, merci d'envoyer un mail á `myodbc@lists.mysql.com` á ce propos !

Avec quelques programmes, vous pouvez avoir l'erreur suivante : `Another user has modifies the record that you have modified`. Dans la plupart des cas, le probléme peut être rèsolu en suivant l'une des mèthodes suivantes :

- ajouter une clef primaire á la table s'il n'en existe pas dèjá une.
- Ajouter une colonne timestamp s'il n'y en a pas dèjá une.
- N'utilisez que des champs rèels doubles. Certains programmes peuvent se bloquer lorsqu'ils comparent des rèels simples.

Si ce qui prècéde ne vous aide pas, vous devez tracer `MyODBC` et essayer de voir pourquoi les choses vont de travers.

8.3.5 Programmes qui fonctionnent avec MyODBC

La majoritè des programmes devraient fonctionner avec `MyODBC`, mais tous ceux que nous avons listès ici sont ceux que nous avons testè nous même, ou dont nous avons reçu la confirmation de la part d'utilisateurs :

Programme
Commentaire

Access Pour le faire fonctionner avec Access :

- Si vous utilisez Access 2000, vous devez installer la derniére Microsoft MDAC (`Microsoft Data Access Components`) (version 2.6 ou plus rècente) depuis `http://www.microsoft.com/data/`. Cela va corriger le bug suivant d'Access : lorsque vous exportez des donnèes vers MySQL, les noms de la table et des colonnes ne sont pas spècifiès. Un autre moyen de pallier á ce bug est de passer en MyODBC version 2.50.33 et MySQL version 3.23.x, qui, ensemble, forment un correctif á ce bug!

  Vous devriez aussi installer le Microsoft Jet 4.0 Service Pack 5 (SP5) disponible á `http://support.microsoft.com/support/kb/articles/Q239/1/14.ASP`. Cela va corriger certains cas ou les colonnes ètaient marquèes comme effacèes (`#deleted#`) dans Access.

  Notez que si vous utilisez MySQL version 3.22, vous devez appliquer le patch MDAC et utiliser MyODBC 2.50.32 ou 2.50.34 et plus rècent pour corriger ce bug.
- Pour toutes les versions d'Access, vous devez activer l'option MyODBC appelèe `Return matching rows`. Pour Access 2.0, vous devez aussi activer `Simulate ODBC 1.0`.
- Vous devez avoir un timestamp dans toutes les tables que vous voulez pouvoir modifier. Pour une meilleure portabilitè, `TIMESTAMP(14)` ou un simple `TIMESTAMP` est recommandè au lieu de `TIMESTAMP(X)`.
- Vous devez avoir une clè primaire dans la table. Sinon, les nouvelles lignes et les lignes modifièes peuvent être marquèes comme effacèes `#DELETED#`.
- N'utilisez que les champs `DOUBLE`. Access èchoue lors des comparaisons entre les champs en simple prècision. Le symptôme est gènèralement que les nouvelles lignes ou les lignes modifièes sont marquèes comme effacèes (`#DELETED#`), ou que vous ne pouvez plus trouver les nouvelles lignes ou les lignes modifièes.
- Si vous reliez des tables via MyODBC, dont l'une des colonnes est de type `BIGINT`, les rèsultats sont affichès comme effacès (`#DELETED`). La solution est :
  - Avoir au moins une colonne de type `TIMESTAMP` comme type de donnèes, et de prèfèrence `TIMESTAMP(14)`.
  - Vèrifiez l'option de connexion `'Change BIGINT columns to INT'` dans l'administrateur ODBC DSN.
  - Effacez le lien de table dans Access, et recrèez le.

Les anciennes lignes seront toujours marquèes comme `#DELETED#`, mais les nouvelles lignes seront affichèes correctement.

- Si vous butez toujours dans l'erreur `Another user has changed your data` aprés avoir ajoutè une colonne `TIMESTAMP`, le truc suivant peut vous aider :

  N'utilisez pas la vue de `table` au format tableur. A la place, crèez un formulaire avec les champs que vous souhaitez, et utilisez ce formulaire comme vue de la table. Vous devez utiliser la propriètè `DefaultValue` pour la colonne `TIMESTAMP`, avec la valeur `NOW()`. C'est une bonne idèe de cacher la colonne `TIMESTAMP` á vos utilisateurs, pour qu'ils ne soient pas perturbès.

- Dans certains cas, Access peut gènèrer des requêtes SQL illègales, que MySQL ne peut comprendre. Vous pouvez corriger cela en sèlectionnant `"Query|SQLSpecific|Pass-Through"` dans le menu Access.
- Access sur NT va indiquer que les colonnes `BLOB` sont des `OLE OBJECTS`. Si vous voulez avoir des colonnes de type `MEMO` á la place, vous devez changer le type de la colonne en `TEXT` avec `ALTER TABLE`.
- Access ne peut pas toujours gèrer correctement les colonnes `DATE`. Sis vous avez des problémes avec, changez ces colonnes en type `DATETIME`.
- Si vous avec une colonne Access de type `BYTE`, Access va essayer de l'exporter sous la forme d'un `TINYINT` au lieu d'un `TINYINT UNSIGNED`. Cela va vous donner des problémes si vous avez des valeurs supèrieures á 127!

ADO

Lorsque vous programmez avec l'API ADO et `MyODBC` vous devez faire attention aux propriètès par dèfaut qui ne sont pas supportèes par MySQL. Par exemple, utiliser la propriètè `CursorLocation Property` comme `adUseServer` va retourner des valeurs de -1 pour `RecordCount Property`. Pour avoir la vèritable valeur, vous devez utiliser la valeur `adUseClient`, comme dans le code VB ci-contre :

```
Dim myconn As New ADODB.Connection
Dim myrs As New Recordset
Dim mySQL As String
Dim myrows As Long

myconn.Open "DSN=MyODBCsample"
mySQL = "SELECT * from user"
myrs.Source = mySQL
Set myrs.ActiveConnection = myconn
myrs.CursorLocation = adUseClient
myrs.Open
myrows = myrs.RecordCount

myrs.Close
myconn.Close
```

Un autre palliatif est d'utiliser la commande `SELECT COUNT(*)`, pour connaître le nombre exact de lignes.

Active server pages (ASP)
: Vous devez utiliser l'option `Return matching rows`.

Applications BDE
: Pour faire fonctionner ces applications, vous devez activer les options `Don't optimize column widths` et `Return matching rows`.

Borland Builder 4
: Lorsque vous dèmarrez une requête, vous pouvez utiliser la propriètè `Active` ou utiliser la mèthode `Open`. Notez que `Active` va commencer en gènèrant automatiquement une requête `SELECT * FROM ...` qui peut ne pas être pratique avec de grosses tables!

ColdFusion (On Unix)
: Les informations suivantes sont reprises de la documentation ColdFusion :

  Utilisez les informations suivantes pour configurer le serveur ColdFusion Server sous Linux pour utiliser le pilote unixODBC avec `MyODBC` pour les serveurs MySQL. Allaire a vèrifiè que `MyODBC` version 2.50.26 fonctionne avec MySQL version 3.22.27 et ColdFusion pour Linux. (toute nouvelle version doit aussi fonctionner). Vous pouvez tèlècharger `MyODBC` á `http://www.mysql.com/downloads/api-myodbc.html`

  ColdFusion version 4.5.1 vous permet d'utiliser l'administrateur ColdFusion pour ajouter des serveurs MySQL. Toutefois, le pilote n'est pas inclus dans ColdFusion version 4.5.1. Auparavant, le pilote MySQL ètait disponible dans le menu dèroulant. Vous devez compiler et copier le pilote `MyODBC` dans '`/opt/coldfusion/lib/libmyodbc.so`'.

  Le dossier Contrib contient le programme '`mydsn-xxx.zip`' qui vous permet de compiler et supprimer des fichiers d'enregistrement DSN pour MyODBC sur les applications Coldfusion.

DataJunction
: Vous devez le modifier pour qu'il exporte des `VARCHAR` plutôt que des `ENUM`, car il exporte ces derniers d'une maniére qui pose des problémes á MySQL.

Excel
: Fonctionne. Quelques conseils :

  - Si vous avez des problémes avec les dates, essayez de les sèlectionner comme des chaînes, en utilisant la fonction `CONCAT()`. Par exemple :

    ```
    select CONCAT(rise_time), CONCAT(set_time)
        from sunrise_sunset;
    ```

    Les valeurs lues sous forme de chaînes seront correctement relues par Excel97.

    Le but de la fonction `CONCAT()` est de faire croire á ODBC que cette colonne est de type "string". Sans la fonction `CONCAT()`, ODBC sait que la colonne est de type heure, et Excel ne le comprendra pas.

    Notez qu'il y a un bug dans Excel, car il convertit automatiquement une chaîne en heure. Cela serait bien si la source ètait un fichier texte, mais

c'est totalement idiot si la source est une connexion ODBC qui fournit les types exacts.

Word

Pour lire des données depuis MySQL vers des documents Word ou Excel, vous devez utiliser le pilote `MyODBC` et l'aide Add-in Microsoft Query.

Par exemple, pour crèer une base de données avec une table contenant 2 colonnes de type texte :

- Insèrez des lignes avec le client `mysql`.
- Crèez un fichier DSN en utilisant le gestionnaire ODBC, par exemple, '`my`' pour la base ci-dessus.
- Ouvrez Word.
- Crèez un nouveau document vide.
- Utilisez la barre "Database", pressez sur le bouton d'insertion.
- Pressez sur le bouton "Get Data".
- Sur la droite de l'ècran, choisissez le bouton "Ms Query".
- Dans ce ècran, crèez une nouvelle source de donnèes avec "New Data Source" et en utilisant le fichier DSN '`my`'.
- Sèlectionner une nouvelle requête.
- Sèlectionnez les colonnes que vous souhaitez.
- Faîtes votre filtre.
- Faîtes votre tri.
- Sèlectionnez "Return Data" de Microsoft Word.
- Cliquez sur "Finish".
- Cliquez sur le bouton "Insert data" et sèlectionnez les lignes.
- Cliquez sur "OK" et vous verrez vous lignes dans votre document Word.

odbcadmin

Programme de test pour ODBC.

Delphi Vous devez utiliser BDE version 3.2 ou plus rècent. Utilisez l'option `Don't optimize column width` lors de la connexion á MySQL.

De plus, voici un code Delphi pratique, pour configurer une entrèe ODBC et une entrèe BDE pour `MyODBC` (l'entrèe BDE requiert le BDE Alias Editor qui est gratuit sur les pages "Delphi Super Page" prés de chez vous. (Merci á Bryan Brunton `bryan@flesherfab.com` pour cela) :

```
fReg:= TRegistry.Create;
  fReg.OpenKey('\Software\ODBC\ODBC.INI\DocumentsFab', True);
  fReg.WriteString('Database', 'Documents');
  fReg.WriteString('Description', ' ');
  fReg.WriteString('Driver', 'C:\WINNT\System32\myodbc.dll');
  fReg.WriteString('Flag', '1');
  fReg.WriteString('Password', '');
  fReg.WriteString('Port', ' ');
```

```
fReg.WriteString('Server', 'xmark');
fReg.WriteString('User', 'winuser');
fReg.OpenKey('\Software\ODBC\ODBC.INI\ODBC Data Sources', True);
fReg.WriteString('DocumentsFab', 'MySQL');
fReg.CloseKey;
fReg.Free;

Memo1.Lines.Add('DATABASE NAME=');
Memo1.Lines.Add('USER NAME=');
Memo1.Lines.Add('ODBC DSN=DocumentsFab');
Memo1.Lines.Add('OPEN MODE=READ/WRITE');
Memo1.Lines.Add('BATCH COUNT=200');
Memo1.Lines.Add('LANGDRIVER=');
Memo1.Lines.Add('MAX ROWS=-1');
Memo1.Lines.Add('SCHEMA CACHE DIR=');
Memo1.Lines.Add('SCHEMA CACHE SIZE=8');
Memo1.Lines.Add('SCHEMA CACHE TIME=-1');
Memo1.Lines.Add('SQLPASSTHRU MODE=SHARED AUTOCOMMIT');
Memo1.Lines.Add('SQLQRYMODE=');
Memo1.Lines.Add('ENABLE SCHEMA CACHE=FALSE');
Memo1.Lines.Add('ENABLE BCD=FALSE');
Memo1.Lines.Add('ROWSET SIZE=20');
Memo1.Lines.Add('BLOBS TO CACHE=64');
Memo1.Lines.Add('BLOB SIZE=32');

AliasEditor.Add('DocumentsFab','MySQL',Memo1.Lines);
```

C++ Builder

Testè avec BDE version 3.0. Le seul probléme connu est que lorsque la structure d'une table change, les champs de requêtes ne changent pas. BDE ne semble pas reconnaître les clès primaires, mais uniquement l'index PRIMARY, même si cela n'a jamais ètè un probléme.

Vision Vous devriez utiliser l'option `Return matching rows`.

Visual Basic

Pour être capable de modifier une table, vous devez dèfinir une clè primaire.

Visual Basic avec ADO ne peut pas gèrer les grands entiers. Cela signifie que les requêtes comme `SHOW PROCESSLIST` ne vont pas fonctionner correctement. Le correctif consiste á ajouter l'option `OPTION=16834` dans la chaîne de connexion ODBC pour d'utiliser l'option `Change BIGINT columns to INT` de MyODBC. Vous pouvez aussi utiliser l'option `Return matching rows`.

VisualInterDev

Si vous rencontrez l'erreur `[Microsoft][ODBC Driver Manager] Driver does not support this parameter`, la raison est que vous avez un grand entier `BIGINT` dans votre rèsultat. Essayez d'utiliser l'option `Change BIGINT columns to INT` de MyODBC.

Visual Objects
: Vous devez utiliser l'option `Don't optimize column widths`.

8.3.6 Comment obtenir la valeur d'une colonne `AUTO_INCREMENT` avec ODBC

Un probléme rècurrent est d'obtenir la derniére valeur gènèrèe automatiquement par une commande `INSERT`. Avec ODBC, vous pouvez procèder de cette façon (en supposons que `auto` est un champ `AUTO_INCREMENT`):

```
INSERT INTO foo (auto,text) VALUES(NULL,'text');
SELECT LAST_INSERT_ID();
```

Ou, si vous voulez juste insèrer cette valeur dans une autre table :

```
INSERT INTO foo (auto,text) VALUES(NULL,'text');
INSERT INTO foo2 (id,text) VALUES(LAST_INSERT_ID(),'text');
```

Voir Section 8.4.6.3 [Getting unique ID], page 659.

Pour quelques applications utilisant ODBC (du moins Delphi et Access), la requête suivante peut être utilisèe pour trouver une ligne insèrèe derniérement :

```
SELECT * FROM nom_de_table WHERE auto IS NULL;
```

8.3.7 Rapporter des problémes avec MyODBC

Si vous rencontrez des difficultès avec `MyODBC`, commencez par faire un fichier de log avec le gestionnaire ODBC (le fichier de log que vous obtenez en demande les logs de ODBCADMIN) et un log `MyODBC`.

Pour obtenir un fichier de log `MyODBC`, vous devez faire ceci :

1. Assurez vous que vous utilisez '`myodbcd.dll`' et non pas '`myodbc.dll`'. Le moyen le plus facile pour le faire est d'obtenir '`myodbcd.dll`' dans la distribution MyODBC et de le copier á la place de '`myodbc.dll`', qui est probablement dans le dossier '`C:\windows\system32`' ou '`C:\winnt\system32`'.

   Notez que vous voudrez probablement rècupèrer votre vieux fichier '`myodbc.dll`' lorsque vous aurez fini de tester, car il est bien plus rapide que '`myodbcd.dll`'.
2. Activez l'option 'Trace MyODBC' dans l'ècran de configuration de `MyODBC`. Le fichier de log sera ècrit dans le fichier '`C:\myodbc.log`'.

   Si l'option de trace n'est pas recommandèe lorsque vous retournez dans l'ècran prècèdent, cela signifie que vous n'utilisez pas `myodbcd.dll` (voir ci-dessus).
3. Dèmarrez votre application, et faîtes la planter.

Vèrifiez le fichier de `trace` `MyODBC`, pour essayer de comprendre ce qui ne va pas. Vous devriez être capable de trouver les requêtes èmises en recherchant la chaîne `>mysql_real_query` dans le fichier '`myodbc.log`'.

Vous devriez aussi essayer de dupiquer la requête dans le client `mysql` ou `admndemo` pour voir si le probléme vient de MyODBC ou MySQL.

Si vous trouvez quelques chose d'incorrect, n'envoyez que les lignes pertinentes (maximum, 40 lignes) á `myodbc@lists.mysql.com`. N'envoyez jamais le fichier de log MyODBC ou ODBC complet!

Si vous êtes incapables de trouver une erreur, la derniére option est de faire une archive (tar ou zip) qui contienne le fichier de trace MyODBC, le fichie de log ODBC, et un fichier README qui contienne une description du probléme. Vous pouvez envoyer le tout á `ftp://support.mysql.com/pub/mysql/secret/`. Seuls nous, á MYSQL AB, pourront accèder á ces fichiers, et nous seront trés diligents avec vos donnèes.

Si vous pouvez crèer un probléme qui reproduit le probléme, essayez de l'uploader aussi!

Si le programem fonctionne avec d'autres serveurs SQL, vous devriez faire un log ODBC oú vous faîtes exactement la même chose sur les autres serveurs SQL.

N'oubliez jamais que plus vous nous fournissez d'explication, plus nous pourront vous aider!

8.4 Interface C pour MySQL

Le code de l'interface C est distribuè avec MySQL. Il est inclus dans la librairie `mysqlclient` et permet aux programmes C d'accèder á une base de donnèes.

Plusieurs des clients de la distribution des sources MySQL sont ècrits en C. Si vous cherchez des exemples de l'utilisation de l'API C, jetez un coup d'oeil á ces clients. Vous pouvez les trouver dans le dossier `clients` dans la distribution des sources de MySQL.

La plupart des autres API clients (toutes á part Java) utilisent la librairie `mysqlclient` pour communiquer avec le serveur MySQL. Cela signifie que, par exemple, vous pouvez tirer partie de plusieurs variables d'environnement utilisèes par d'autres programmes clients, car elles sont rèfèrencèes dans la librairie. Voyez Section 4.8 [Client-Side Scripts], page 305, pour une liste de ces variables.

Le client a une limite au niveau de la taille du tampon de communication. Sa taille, qui est initialement allouèe (16K octets), est automatiquement augmentèe á la taille maximale (le maximum est de 16M). Puisque la taille des tampons n'est augmentèe qu'á la demande, le fait d'augmenter la limite par dèfaut n'utilisera pas plus de ressources. La vèrification de la taille est plus une vèrification pour les requêtes et les paquets de communication èrronès.

Le tampon de communication doit être assez grand pour pouvoir contenir une requête SQL unique (pour le traffic client-vers-serveur) et une ligne de donnèes retournèes (pour le traffic serveur-vers-client). Chaque thread du tampon de communication est dynamiquement agrandit pour gèrer les requêtes ou les donnèes atteignant la taille limite. Par exemple, si vous avez un champ de type `BLOB` qui contient des donnèes de 16M de taille, vous devez avoir un tampon de communication qui peut au moins atteindre 16 M en taille (côtè client et serveur). Le maximum par dèfaut du client est 16M, mais le maximum par dèfaut du serveur est 1M. Vous pouvez l'augmenter en changeant la valeur du paramétre `max_allowed_packet` quand le serveur est dèmarrè. Voir Section 5.5.2 [Server parameters], page 390.

Le serveur MySQL rèduit automatiquement chaque tampon de communication á `net_buffer_length` octets aprés chaque requête. Pour les clients, la taille du tampon associè avec une connexion n'est pas rèduite tant que la connexion n'est pas fermèe. Au moment de la fermeture, la mèmoire du client est rèclammèe.

Pour programmer avec les threads, voyez Section 8.4.8 [Threaded clients], page 660. Pour crèer une application indèpendante qui inclut á la fois le serveur et le client dans le même programme (et ne communique pas avec un serveur MySQL externe), voyez Section 8.4.9 [libmysqld], page 662.

8.4.1 Types de données de l'API C

`MYSQL` Cette structure reprèsente un gestionnaire de connexion á la base de donnèes. Elle est utilisèe dans la plupart des fonctions MySQL.

`MYSQL_RES`
Cette structure reprèsente le rèsultat d'une requête qui retourne des lignes (`SELECT`, `SHOW`, `DESCRIBE`, `EXPLAIN`). L'information retournèe par une requête est appelèe *jeu de rèsultats* dans le reste de cette section.

`MYSQL_ROW`
C'est une reprèsentation sûre pour les types d'une ligne de donnèes. Elle est actuellement implèmentèe en tant que tableau de chaîne á octets comptès. (Vous ne pouvez la traiter en tant que chaîne terminèe par une valeur nulle si les valeurs du champ peuvent contenir des donnèes binaires, car de telles valeurs peuvent contenir elles-même des octets nuls.) Les lignes sont obtenues en appelant `mysql_fetch_row()`.

`MYSQL_FIELD`
Cette structure contient des informations á propos du champ, tel que son nom, son type, et sa taille. Ses membres sont dècrit en plus de dètails ici. Vous pouvez obtenir une structure `MYSQL_FIELD` pour chaque champ en appelant plusieurs fois `mysql_fetch_field()`. Les valeurs des champs ne font pas partie de la structure; elles sont contenues dans une structure `MYSQL_ROW`.

`MYSQL_FIELD_OFFSET`
C'est une reprèsentation sûre des types pour les index dans une liste de champs MySQL. (Utilisès par `mysql_field_seek()`.) Les index sont des numèros de champs, dans une ligne, commençant á zèro.

`my_ulonglong`
Le type utilisè pour le nombre de lignes et pour `mysql_affected_rows()`, `mysql_num_rows()`, et `mysql_insert_id()`. Ce type fournit une echelle allant de `0` á `1.84e19`.

Sur quelques systémes, essayer d'ècrire la valeur d'un type `my_ulonglong` ne fonctionnera pas. Pour ècrire une telle valeur, convertissez lá en `unsigned long` et utilisez un format d'impression `%lu`. Exemple :

```
printf ("Nombre de lignes : %lu\n", (unsigned long) mysql_num_rows(result
```

La structure `MYSQL_FIELD` contient les membres listès ici :

`char * name`
Le nom du champ, une chaîne terminèe par une valeur nulle.

`char * table`
Le nom de la table contenant ce champ, s'il n'est pas calculè. Pour les champs calculès, la valeur de `table` est une chaîne vide.

`char * def`
La valeur par dèfaut de ce champ, en tant que chaîne terminèe par une valeur nulle. Ce n'est dèfinit que si vous utilisez `mysql_list_fields()`.

`enum enum_field_types type`

Le type du champ. La valeur de `type` peut être l'une des suivantes :

Valeur de type	Description du type
`FIELD_TYPE_TINY`	Champ `TINYINT`
`FIELD_TYPE_SHORT`	Champ `SMALLINT`
`FIELD_TYPE_LONG`	Champ `INTEGER`
`FIELD_TYPE_INT24`	Champ `MEDIUMINT`
`FIELD_TYPE_LONGLONG`	Champ `BIGINT`
`FIELD_TYPE_DECIMAL`	Champ `DECIMAL` ou `NUMERIC`
`FIELD_TYPE_FLOAT`	Champ `FLOAT`
`FIELD_TYPE_DOUBLE`	Champ `DOUBLE` ou `REAL`
`FIELD_TYPE_TIMESTAMP`	Champ `TIMESTAMP`
`FIELD_TYPE_DATE`	Champ `DATE`
`FIELD_TYPE_TIME`	Champ `TIME`
`FIELD_TYPE_DATETIME`	Champ `DATETIME`
`FIELD_TYPE_YEAR`	Champ `YEAR`
`FIELD_TYPE_STRING`	Champ chaîne de caractéres (`CHAR` ou `VARCHAR`)
`FIELD_TYPE_BLOB`	Champ `BLOB` ou `TEXT` (utilisez `max_length` pour dèterminer la taille maximale)
`FIELD_TYPE_SET`	Champ `SET`
`FIELD_TYPE_ENUM`	Champ `ENUM`
`FIELD_TYPE_NULL`	Champ de type `NULL`
`FIELD_TYPE_CHAR`	Dèsapprouvè, utilisez `FIELD_TYPE_TINY` á la place

Vous pouvez utiliser la macro `IS_NUM()` pour tester si un champ est de type numèrique ou non. Passez la valeur de `type` á `IS_NUM()` et elle sera èvaluèe á TRUE si le champ est numèrique :

```
if (IS_NUM(field->type))
    printf("Le champ est numèrique\n");
```

`unsigned int length`

La taille du champ, comme spècifiè dans la dèfinition de la table.

`unsigned int max_length`

La longueur maximale du champ pour le jeu de rèsultats (la taille de la plus longue valeur de champ actuellement dans le jeu de rèsultat). Si vous utilisez `mysql_store_result()` ou `mysql_list_fields()`, cela contient la longueur maximale pour le champ. Si vous utilisez `mysql_use_result()`, la valeur de cette variable est zèro.

`unsigned int flags`

Les diffèrents attributs sous forme de bits pour le champ. La valeur de `flags` peut avoir zèro ou plusieurs de ces bits suivants activès :

Valeur d'attribut	Description d'attribut
`NOT_NULL_FLAG`	Le champ ne peut être `NULL`
`PRI_KEY_FLAG`	Le champ fait parti d'une clef primaire
`UNIQUE_KEY_FLAG`	Le champ fait parti d'une clef unique
`MULTIPLE_KEY_FLAG`	Le champ fait parti d'une clef non-unique

UNSIGNED_FLAG	Le champ posséde l'attribut UNSIGNED
ZEROFILL_FLAG	Le champ posséde l'attribut ZEROFILL
BINARY_FLAG	Le champ posséde l'attribut BINARY
AUTO_INCREMENT_FLAG	Le champ posséde l'attribut AUTO_INCREMENT
ENUM_FLAG	Le champ est un ENUM (dèsapprouvè)
SET_FLAG	Le champ est un SET (dèsapprouvè)
BLOB_FLAG	Le champ est un BLOB ou TEXT (dèsapprouvè)
TIMESTAMP_FLAG	Le champ est un TIMESTAMP (dèsapprouvè)

L'utilisation des attributs BLOB_FLAG, ENUM_FLAG, SET_FLAG, et TIMESTAMP_FLAG est dèsapprouvè car ils indiquent un type de champ plutôt qu'un attribut de type de champ. Il est prèfèrable de tester field->type avec FIELD_TYPE_BLOB, FIELD_TYPE_ENUM, FIELD_TYPE_SET, ou FIELD_TYPE_TIMESTAMP á la place.

L'exemple suivant illustre une utilisation typique de la valeur de flags :

```
if (field->flags & NOT_NULL_FLAG)
    printf("Le champ ne peut être nul\n");
```

Vous pouvez utiliser les diffèrentes macros ci-dessous pour dèterminer le status boolèen de la valeur de l'attribut :

Status de l'attribut	Description
IS_NOT_NULL(flags)	Vrai si le champ est dèfini en tant que NOT NULL
IS_PRI_KEY(flags)	Vrai si le champ est une clef primaire
IS_BLOB(flags)	Vrai si le champ est un BLOB ou TEXT (dèsapprouvè; tester plutôt field->type)

unsigned int decimals

Le nombre de dècimales pour les champs numèriques.

8.4.2 Vue d'ensemble des fonctions de l'API C

Les fonctions disponibles dans l'API C sont listèes ici et dècrites en plus de dètails dans la section suivante. Voir Section 8.4.3 [C API functions], page 616.

Fonction	Description
mysql_affected_rows()	Retourne le nombre de lignes changèes/effacèes/insèrès par le dernier UPDATE, DELETE, ou INSERT.
mysql_change_user()	Change l'utilisateur et la base de donnèes pour une connexion ouverte.
mysql_character_set_name()	Retourne le nom du jeu de caractére de la connexion.
mysql_close()	Ferme une connexion au serveur.
mysql_connect()	Connecte á un serveur MySQL. Cette fonction est dèsapprouvèe; utilisez mysql_real_connect() á la place.
mysql_create_db()	Crèe une base de donnèes. Cette fonction est dèsapprouvèe, utilisez plutôt la commande SQL CREATE DATABASE.

mysql_data_seek()	Dèplace le pointeur vers une ligne arbitraire dans le jeu de rèsultats de la requête.
mysql_debug()	Effectue un `DBUG_PUSH` avec la chaîne donnèe.
mysql_drop_db()	Supprime une base de donnèes. Cette fonction est dèsapprouvèe, utilisez plutôt la commande SQL `DROP DATABASE`.
mysql_dump_debug_info()	Demande au serveur d'ècrire les informations de dèbogage dans un fichier de log.
mysql_eof()	Dètermine si la derniére ligne du jeu de rèsultats a ètè lue ou non. Cette fonction est dèsapprouvèe, vous pouvez utiliser `mysql_errno()` ou `mysql_error()` á la place.
mysql_errno()	Retourne le numèro de l'erreur de la fonction appelèe en dernier.
mysql_error()	Retourne le message d'erreur de la derniére fonction MySQL appelèe.
mysql_escape_string()	Protége une chaîne en èchappant les caractéres spèciaux.
mysql_fetch_field()	Retourne le type du champ suivant dans la table.
mysql_fetch_field_direct()	Retourne le type d'une colonne, ètant donnè un numèro de champ.
mysql_fetch_fields()	Retourne un tableau avec toutes les structures de champs.
mysql_fetch_lengths()	Retourne la longueur de toutes les colonnes dans la ligne suivante.
mysql_fetch_row()	Rècupére la ligne suivante dans le jeu de rèsultats.
mysql_field_seek()	Place le curseur de colonne sur une colonne prècise.
mysql_field_count()	Retourne le nombre de colonnes dans le rèsultat pour la requête la plus rècente.
mysql_field_tell()	Retourne la position du curseur de champs utilisè pour le dernier appel á `mysql_fetch_field()`.
mysql_free_result()	Libére la mèmoire utilisèe par un jeu de rèsultats.
mysql_get_client_info()	Retourne la version du client.
mysql_get_host_info()	Retourne une chaîne dècrivant la connexion.
mysql_get_proto_info()	Retourne la version du protocole utilisè par la connexion.
mysql_get_server_info()	Retourne la version du serveur.

mysql_info()	Retourne des informations á propos de la requête la plus rècente.
mysql_init()	Rècupére ou initialise une structure `MYSQL`.
mysql_insert_id()	Retourne l'identifiè gènèrè pour une colonne `AUTO_INCREMENT` par la derniére requête.
mysql_kill()	Termine un processus donnè.
mysql_list_dbs()	Retourne les noms des bases de donnèes rèpondant á une expression règuliére simple.
mysql_list_fields()	Retourne les noms des champs rèpondants á une expression règuliére simple.
mysql_list_processes()	Retourne une liste des processus courants du serveur.
mysql_list_tables()	Retourne les noms des tables rèpondants á une expression règuliére simple.
mysql_num_fields()	Retourne le nombre de colonnes dans un jeu de rèsultats.
mysql_num_rows()	Retourne le nombre de lignes dans le jeu de rèsultats.
mysql_options()	Configure les options de connexion pour `mysql_connect()`.
mysql_ping()	Vèrifie si la connexion au serveur a toujours lieu. Reconnecte si besoin.
mysql_query()	Exècute une requête SQL spècifèe en tant que chaîne terminèe par un caractére nul.
mysql_real_connect()	Connecte á un serveur MySQL.
mysql_real_escape_string()	Protége les caractéres spèciaux dans une chaîne utilisable dans une requête SQL, en prenant en compte le jeu de caractéres courant de la connexion.
mysql_real_query()	Exècute une requête SQL spècifièe en tant que chaîne comptèe.
mysql_reload()	Demande au serveur de recharger la table des droits.
mysql_row_seek()	Dèplace le pointeur vers un ligne dans le jeu de rèsultats, en utilisant une valeur retournèe par `mysql_row_tell()`.
mysql_row_tell()	Retourne la position du curseur de lignes.
mysql_select_db()	Sèlectionne une base de donnèes.
mysql_shutdown()	Termine le serveur de base de donnèes.
mysql_stat()	Retourne le statut du serveur dans une chaîne.

mysql_store_result()	Rècupére le jeu de rèsulats complet dans le client.
mysql_thread_id()	Retourne l'identifiant du thread courant.
mysql_thread_safe()	Retourne 1 si le client est compilè en tant que thread-safe.
mysql_use_result()	Initialise une rècupèration ligne par ligne des rèsultats.

Pour vous connecter au serveur, appelez `mysql_init()` pour initialiser un gestionnaire de connexion, puis appelez `mysql_real_connect()` avec ce gestionnaire (avec d'autres informations tel que l'hôte, l'utilisateur et le mot de passe). Lors de la connexion, `mysql_real_connect()` dèfinit l'option `reconnect` (quit fait partie de la structure MYSQL) á `1`. Cette option indique, dans le cas oú une requête ne peut être exècutèe á cause d'une perte de connexion, d'essayer de se reconnecter au serveur avant d'abandonner. Lorsque vous n'avez plus besoin de la connexion, appelez `mysql_close()` pour la clore.

Tant qu'une connexion est active, le client envoi des requêtes SQL au serveur á l'aide de `mysql_query()` ou `mysql_real_query()`. La diffèrence entre les deux est que `mysql_query()` s'attend á ce que la requête soit spècifiée en tant que chaîne terminée par la chaîne nulle, tandis que `mysql_real_query()` attend une chaîne de longueur connue. Si la chaîne contient des donnèes binaires (incluant l'octet nul), vous devez utiliser `mysql_real_query()`.

Pour chaque requête non-sèlective (par exemple, `INSERT`, `UPDATE`, `DELETE`), vous pouvez trouver combien de lignes ont ètè mises á jour (affectèes) en appelant `mysql_affected_rows()`.

Pour les requêtes `SELECT`, vous rècupèrez les lignes sèlectionnèes dans un jeu de rèsultat. (Notez que quelques commandes ont le même comportement que `SELECT`, dans le sens oú elle renvoient des lignes. Cela inclut `SHOW`, `DESCRIBE`, et `EXPLAIN`. Elles doivent être traitèes de la même façon que les requêtes `SELECT`.)

Il y'a deux façons pour un client de gèrer les jeux de rèsultats. Une mèthode consiste á rècupèrer le jeu de rèsultat en entier et en une seule fois en appelant `mysql_store_result()`. Cette fonction obtient toutes les lignes retournèes par la requête et les stocke dans le client. La seconde mèthode consiste á initialiser une rècupèration ligne-par-ligne du jeu de rèsultats en appelant `mysql_use_result()`. Cette fonction initie la rècupèration, mais ne rècupére actuellement aucune ligne á partir du serveur.

Dans les deux cas, vous accèdez aux ligne en appelant `mysql_fetch_row()`. Avec `mysql_store_result()`, `mysql_fetch_row()` accéde aux lignes qui ont dèjá ètè rècupèrèes á partir du serveur. Avec `mysql_use_result()`, `mysql_fetch_row()` rècupére actuellement la ligne á partir du serveur. Les informations á propos de la taille des donnèes dans chaque ligne est disponible en appelant `mysql_fetch_lengths()`.

Aprés avoir fini de traiter le jeu de rèsultats, appelez `mysql_free_result()` pour libèrer la mèmoire utilisèe.

Les deux mècanismes de rècupèration sont complèmentaires. Les programmes clients doivent utiliser l'approche qui leur convient le mieux. En pratique, les clients tendent plus á utiliser `mysql_store_result()`.

Un avantage de `mysql_store_result()` est que puisque toutes les lignes ont ètè rècupèrèes dans le client, vous ne pouvez pas que accèder aux lignes sèquentiellement, vous pouvez revenir en arriére ou avancer dans le jeu de rèsultats en utilisant `mysql_data_seek()` ou

`mysql_row_seek()` pour changer la position de la ligne courante dans le jeu de rèsultats. Vous pouvez aussi trouver le nombre total des lignes en appelant `mysql_num_rows()`. D'un autre côtè, les besoins en mèmoire de `mysql_store_result()` peuvent être trés grands pour les grands jeux de rèsultats et vous aurez des chances de rencontrer des conditions out-of-memory.

Un avantage de `mysql_use_result()` est que le client a besoinde moins de mèmoire pour le jeu de rèsultats car il utilise une ligne á la fois (et puisque il y'a moins de pertes de mèmoire, `mysql_use_result()` peut être plus rapide). Les inconvènients sont que vous devez rècupèrer chaque ligne rapidement pour èviter de bloquer le serveur, vous n'avez pas d'accés alèatoires aux lignes dans le jeu de rèsultats (vous ne pouvez accèder aux lignes que sèquentiellement), et vous ne savez pas combien de lignes comporte le jeu de rèsultats tant que vous ne les avez pas toutes rècupèrèes. De plus, vous **devez** rècupèrer toutes les lignes même si vous trouvez entre-temps l'informations que vous cherchiez.

L'API permet aux clients de gèrer correctement les requêtes (rècupèrant les lignes seulement en cas de besoin) sans savoir si la requête ètait un `SELECT` ou non. Vous pouvez faire cela en appelant `mysql_store_result()` aprés chaque `mysql_query()` (ou `mysql_real_query()`). Si l'appel au jeu de rèsultats rèussi, la requête ètait un `SELECT` et vous pouvez lire les lignes. Sinon, appelez `mysql_field_count()` pour vèrifier si un rèsultat aurait du être retournè. Si `mysql_field_count()` retourne zèro, la requête n'a pas retournè de donnèes (cela indique que c'ètait un `INSERT`, `UPDATE`, `DELETE`, etc.), et ne devait pas retourner de lignes. Si `mysql_field_count()` est non-nul, la requête aurait du retourner des lignes, mais ne l'a pas fait. Cela indique que la requête ètait un `SELECT` qui a èchouè. Reportez vous á la description de `mysql_field_count()` pour un exemple d'utilisation.

`mysql_store_result()` et `mysql_use_result()` vous permettent d'obtenir des informations á propos des champs qui constituent le jeu de rèsultat (le nombre de champs, leurs noms et types, etc.). Vous pouvez accèder aux informations du champ sèquentiellement dans une ligne en appelant plusieurs fois `mysql_fetch_field()`, ou avec le numèro du champ dans la ligne en appelant `mysql_fetch_field_direct()`. La position courante du pointeur de champ peut être changèe en appelant `mysql_field_seek()`. Changer le pointeur de champ affecte les appels suivants á `mysql_fetch_field()`. Vous pouvez aussi obtenir en une seule fois les informations sur les champs en appelant `mysql_fetch_fields()`.

Pour dètecter les erreurs, MySQL fournit un accés aux informations des erreurs via les fonctions `mysql_errno()` et `mysql_error()`. Elles retournent le code de l'erreur et le message pour la derniére fonction invoquèe qui aurait pu rèussir ou èchouer, vous permettant ainsi de dèterminer les erreurs et leurs causes.

8.4.3 Description des fonctions de l'API C

Dans les descriptions suivantes, un paramétre ou retour de fonction `NULL` correspond au `NULL` dans le sens C du terme, et non dans le sens de la valeur `NULL` de MySQL.

Les fonctions qui retournent une valeur retournent la plupart du temps un pointeur ou un entier. Sauf en cas d'indications contraires, les fonctions retournant un pointeur retournent une valeur non-`NULL` pour indiquer un succés ou une valeur `NULL` pour indiquer une erreur, les fonctions retournant un entier retournent zèro pour indiquer un succés et une valeur non-nulle en cas d'erreur. Notez que "non-nulle" ne signifie rien de plus que cela. Sauf si la description de la fonction le dit, ne testez pas avec une valeur diffèrente de zèro :

```
if (result)                    /* correct */
    ... erreur ...

if (result < 0)                /* incorrect */
    ... erreur ...

if (result == -1)              /* incorrect */
    ... erreur ...
```

Lorsqu'une fonction retourne une erreur, la section **Erreurs** du descriptif de la fonction liste les types d'erreurs possibles. Vous pouvez trouver celle qui est arrivèe en appelant `mysql_errno()`. Une chaîne de caractéres reprèsentant l'erreur peut être obtenue en appelant `mysql_error()`.

8.4.3.1 mysql_affected_rows()

`my_ulonglong mysql_affected_rows(MYSQL *mysql)`

Description

Retourne le nombre de lignes modifièes par la derniére commande `UPDATE`, supprimèes par la derniére commande `DELETE` ou insèrèe par la derniére commande `INSERT`. Peut être appelèe immèdiatement aprés `mysql_query()` pour les commandes `UPDATE`, `DELETE`, ou `INSERT`. Pour la commande `SELECT`, `mysql_affected_rows()` fonctionne comme `mysql_num_rows()`.

Valeur de retour

Un entier supèrieur á zèro indique le nombre de lignes affectèes ou sèlectionnèes. Zèro indique qu'aucun enregistrement n'a ètè mis á jour pour une requête `UPDATE`, qu'aucune lignes n'a correspondu á la clause `WHERE` dans la requête ou que celle ci n'a pas encore ètè exècutèe. -1 indique que la requête a renvoyè une erreur ou que, pour une requête `SELECT`, `mysql_affected_rows()` a ètè appelèe avant `mysql_store_result()`.

Erreurs

Aucune.

Exemple

```
mysql_query(&mysql,"UPDATE products SET cost=cost*1.25 WHERE group=10");
printf("%ld produits mis á jour",(long) mysql_affected_rows(&mysql));
```

Si on spècifie l'option `CLIENT_FOUND_ROWS` en se connectant á `mysqld`, `mysql_affected_rows()` retournera le nombre d'enregistrements correspondants á la clause `WHERE` pour une requête `UPDATE`.

Notez que quand on utilise une commande `REPLACE`, `mysql_affected_rows()` retournera 2 si le nouvel enregistrement en a remplacè un ancien. 2 en retour car dans ce cas, l'ancienne ligne a ètè supprimè puis la nouvelle insèrèe.

8.4.3.2 mysql_change_user()

```
my_bool mysql_change_user(MYSQL *mysql, const char *user, const char
*password, const char *db)
```

Description

Change l'utilisateur et dèfinit la base de donnèes spècifièe par `db` en tant que base de donnèes par dèfaut (courante) dans la connexion spècifièe par `mysql`. Pour les requêtes suivantes, cette base de donnèes sera celle utilisèe pour les rèfèrences aux tables ne spècifiant pas explicitement une base de donnèes.

Cette fonction a ètè introduite á la version 3.23.3 de MySQL.

`mysql_change_user()` èchoue si l'utilisateur ne peut être authentifiè ou s'il n'a pas le droit d'utiliser cette base de donnèes. Dans ce cas, l'utilisateur et la base de donnèes ne sont pas changès.

Le paramétre `db` peut être mis á `NULL` si vous ne voulez pas avoir de base de donnèes par dèfaut.

Valeur de retour

Zèro en cas de succés. Diffèrent de zèro si une erreur se produit.

Erreurs

Les mêmes que vous pouvez obtenir avec `mysql_real_connect()`.

`CR_COMMANDS_OUT_OF_SYNC`
: Les commandes n'ont pas ètè exècutèes dans le bon ordre.

`CR_SERVER_GONE_ERROR`
: Le serveur MySQL ne rèponds pas.

`CR_SERVER_LOST`
: La connexion au serveur a ètè perdue au cours la requête.

`CR_UNKNOWN_ERROR`
: Une erreur inconnue s'est produite.

`ER_UNKNOWN_COM_ERROR`
: Le serveur MySQL n'implèmente pas cette commande (probablement un ancien serveur)

`ER_ACCESS_DENIED_ERROR`
: L'utilisateur ou le mot de passe ètaient erronès.

`ER_BAD_DB_ERROR`
La base de donnèes n'existe pas.

`ER_DBACCESS_DENIED_ERROR`
L'utilisateur n'a pas le droit d'accèder á la base de donnèes.

`ER_WRONG_DB_NAME`
Le nom de la base de donnèes ètait trop long.

Exemple

```
if (mysql_change_user(&mysql, "user", "password", "new_database"))
{
   fprintf(stderr, "Impossible de changer d'utilisateur.  Erreur : %s\n",
           mysql_error(&mysql));
}
```

8.4.3.3 mysql_character_set_name()

```
const char *mysql_character_set_name(MYSQL *mysql)
```

Description

Retourne le jeu de caractéres par dèfaut de la connexion courante.

Valeur de retour

Le jeu de caractéres par dèfaut

Erreurs

Aucune.

8.4.3.4 mysql_close()

```
void mysql_close(MYSQL *mysql)
```

Description

Ferme la connexion ouverte prècèdemment. `mysql_close()` dèsalloue aussi le pointeur de connexion pointè par `mysql`, si celui ci avait ètè allouè dynamiquement par `mysql_init()` ou `mysql_connect()`.

Valeur de retour

Aucune.

Erreurs

Aucune.

8.4.3.5 mysql_connect()

```
MYSQL *mysql_connect(MYSQL *mysql, const char *host, const char *user, const
char *passwd)
```

Description

Cette fonction est dèsapprouvèe. Il est prèfèrable d'utiliser `mysql_real_connect()` á la place.

`mysql_connect()` essaye d'ètablir une connexion á un serveur MySQL lancè sur `host`. `mysql_connect()` doit s'achever avec succés avant que vous ne puissiez exècuter l'une des autres fonctions de l'API, á l'exception de `mysql_get_client_info()`.

La signification des paramétres est la même que pour ceux de la fonction `mysql_real_connect()` á la diffèrence que le paramétre de connexion peut être `NULL`. Dans ce cas, l'API C alloue automatiquement une mèmoire pour la structure de connexion et la libére quand vous appelez `mysql_close()`. Le dèsavantage de cette approche est que vous ne pouvez pas rècupèrer les messages d'erreur si la connexion èchoue. (Pour obtenir des informations sur les erreurs á partir de `mysql_errno()` ou `mysql_error()`, vous devez fournir un pointeur `MYSQL` valide.)

Valeur de retour

La même que pour `mysql_real_connect()`.

Erreurs

Les mêmes que pour `mysql_real_connect()`.

8.4.3.6 mysql_create_db()

```
int mysql_create_db(MYSQL *mysql, const char *db)
```

Description

Crèe la base de donnèes nommèe avec le paramétre `db`.

Cette fonction est dèsapprouvèe. Il est prèfèrable d'utiliser `mysql_query()` pour gènèrer une requête SQL `CREATE DATABASE` á la place.

Valeur de retour

Zèro si la base á ètè crèe avec succés. Diffèrente de zèro si une erreur est survenue.

Erreurs

`CR_COMMANDS_OUT_OF_SYNC`
: Les commandes n'ont pas ètè exècutèes dans le bon ordre.

`CR_SERVER_GONE_ERROR`
: Le serveur MySQL ne rèponds pas.

`CR_SERVER_LOST`
: La connexion au serveur a ètè perdue au cours la requête.

`CR_UNKNOWN_ERROR`
: Une erreur inconnue s'est produite.

Exemple

```
if(mysql_create_db(&mysql, "ma_base"))
{
   fprintf(stderr, "Impossible de crèer une nouvelle base de donnèes. Erreur : %s\n
           mysql_error(&mysql));
}
```

8.4.3.7 `mysql_data_seek()`

`void mysql_data_seek(MYSQL_RES *result, my_ulonglong offset)`

Description

Se dèplace vers une ligne arbitraire d'un jeu de rèsultat de requête. Cela nècessite que la structure du jeu de rèsultat contienne la totalitè du rèsultat de la requête, de ce fait `mysql_data_seek()` peut être utilisèe en conjonction avec `mysql_store_result()`, mais pas avec `mysql_use_result()`.

L'index de la ligne doit être compris entre 0 et `mysql_num_rows(result)-1`.

Valeur de retour

Aucune.

Erreurs

Aucune.

8.4.3.8 `mysql_debug()`

`void mysql_debug(const char *debug)`

Description

Provoque un `DBUG_PUSH` avec la chaîne donnèe. `mysql_debug()` utilises la librairie de dèbogage Fred Fish. Pour utiliser cette fonction vous devez compiler la librairie client avec le support dèbogage. Voir Section E.1 [Debugging server], page 815. Voir Section E.2 [Debugging client], page 821.

Valeur de retour

Aucune.

Erreurs

Aucune.

Exemple

L'appel montrè ici fais gènèrer á la librairie du client un fichier de trace dans '`/tmp/client.trace`' sur la machine du client :

```
mysql_debug("d:t:O,/tmp/client.trace");
```

8.4.3.9 mysql_drop_db()

```
int mysql_drop_db(MYSQL *mysql, const char *db)
```

Description

Supprime la base de donnèes nommèe avec le paramétre `db`.

Cette fonction est dèsapprouvèe. Il est prèfèrable d'utiliser `mysql_query()` pour gènèrer une requête SQL `DROP DATABASE` á la place.

Valeur de retour

Zèro si la base á ètè effacèe avec succés. Diffèrente de zèro si une erreur est survenue.

Erreurs

`CR_COMMANDS_OUT_OF_SYNC`
: Les commandes n'ont pas ètè exècutèes dans le bon ordre.

`CR_SERVER_GONE_ERROR`
: Le serveur MySQL ne rèponds pas.

`CR_SERVER_LOST`
: La connexion au serveur a ètè perdue au cours la requête.

`CR_UNKNOWN_ERROR`
: Une erreur inconnue s'est produite.

Exemple

```
if(mysql_drop_db(&mysql, "ma_base"))
  fprintf(stderr, "Impossible de supprimer la base de donnèes. Erreur : %s\n",
          mysql_error(&mysql));
```

8.4.3.10 mysql_dump_debug_info()

```
int mysql_dump_debug_info(MYSQL *mysql)
```

Description

Demande au serveur d'ècrire quelques informations de dèbogage dans le log. Pour que cela fonctionne, il faut que l'utilisateur ait le droit `SUPER`.

Valeur de retour

Zèro si la commande a ètè effectuèe avec succés. Diffèrente de zèro si une erreur est survenue.

Erreurs

`CR_COMMANDS_OUT_OF_SYNC`
: Les commandes n'ont pas ètè exècutèes dans le bon ordre.

`CR_SERVER_GONE_ERROR`
: Le serveur MySQL ne rèponds pas.

`CR_SERVER_LOST`
: La connexion au serveur a ètè perdue au cours la requête.

`CR_UNKNOWN_ERROR`
: Une erreur inconnue s'est produite.

8.4.3.11 mysql_eof()

```
my_bool mysql_eof(MYSQL_RES *result)
```

Description

Cette fonction est dèsapprouvèe. Vous pouvez utiliser `mysql_errno()` ou `mysql_error()` á la place.

`mysql_eof()` dètermine si la derniére ligne d'un jeu de rèsultats a ètè lue.

Si vous obtenez un jeu de rèsultats suite á un appel á `mysql_store_result()`, le client reçois le jeu entier en une seule opèration. Dans ce cas, un retour `NULL` de la fonction `mysql_fetch_row()` signifie toujours que la fin du jeu de rèsultat a ètè atteinte et il n'est donc pas nècessaire d'appeler `mysql_eof()`. Lors d'une utilisation avec `mysql_store_result()`, `mysql_eof()` retournera toujours true.

D'un autre côtè, si vous utilisez `mysql_use_result()` pour initialiser la rècupèration d'un jeu de rèsultats, les lignes sont obtenues du serveur une par une lors des appels successifs de `mysql_fetch_row()`. Puisque une erreur peut survenir á la connexion durant ce processus, une valeur de retour `NULL` de la part de `mysql_fetch_row()` ne signifie pas nècessairement que la fin du jeu de rèsultats a ètè atteinte normalement. Dans ce cas, vous pouvez utiliser `mysql_eof()` pour determiner ce qui est arrivè. `mysql_eof()` retourne une valeur non-nulle si la fin du jeu de rèsultats a ètè atteinte et zèro en cas d'erreur.

Historiquement, `mysql_eof()` a vu le jour avant les fonctions d'erreurs standards de MySQL `mysql_errno()` et `mysql_error()`. Puisque ces fonctions fournissent les mêmes informations, leur utilisation est prèfèrèe á `mysql_eof()`, qui est maintenant dèsapprouvèe. (En fait, elles fournissent plus d'informations, car `mysql_eof()` ne retourne que des valeurs boolèennes alors que les fonctions d'erreurs indiquent les raisons des erreurs lorsqu'elles surviennent.)

Valeur de retour

Zèro si aucune erreur n'est survenue. Autre chose dans le cas contraire.

Erreurs

Aucune.

Exemple

L'exemple suivant vous montre comment vous devez utiliser `mysql_eof()`:

```
mysql_query(&mysql,"SELECT * FROM une_table");
result = mysql_use_result(&mysql);
while((row = mysql_fetch_row(result)))
{
    // traite les donnèes
}
if(!mysql_eof(result))  // mysql_fetch_row() failed due to an error
{
    fprintf(stderr, "Erreur : %s\n", mysql_error(&mysql));
}
```

Vous pouvez reproduire la même chose avec les fonctions d'erreurs de MySQL :

```
mysql_query(&mysql,"SELECT * FROM une_table");
result = mysql_use_result(&mysql);
while((row = mysql_fetch_row(result)))
{
    // traite les donnèes
}
if(mysql_errno(&mysql))  // mysql_fetch_row() ne marche pas á cause d'une erreur
{
    fprintf(stderr, "Erreur : %s\n", mysql_error(&mysql));
}
```

8.4.3.12 mysql_errno()

```
unsigned int mysql_errno(MYSQL *mysql)
```

Description

Pour la connexion spècifiée par `mysql`, `mysql_errno()` retourne le code de l'erreur pour l'appel le plus rècent á une fonction de l'API qui peut rèussir ou èchouer. Un zèro en valeur de retour signifie qu'aucune erreur ne s'est produite. Les codes erreur du client sont listès dans le fichier d'entête MySQL (‘`errmsg.h`’). Les codes erreur du serveur sont listès dans le fichier ‘`mysqld_error.h`’. Dans les sources de la distribution MySQL vous pouvez trouver la liste compléte des messages d'erreur et le code qui leur est associè dans le fichier ‘`Docs/mysqld_error.txt`’.

Valeur de retour

Un code d'erreur. Zèro si aucune erreur n'est survenue.

Erreurs

Aucune.

8.4.3.13 mysql_error()

```
char *mysql_error(MYSQL *mysql)
```

Description

Pour la connexion spècifiée par `mysql`, `mysql_error()` retourne le message d'erreur pour l'appel le plus rècent á une fonction de l'API qui peut rèussir ou èchouer. Une chaîne vide (`""`) est retournèe si aucune erreur n'est survenue. Cela signifie que les deux tests suivants sont èquivalents :

```
if(mysql_errno(&mysql))
{
    // une erreur est survenue
}

if(mysql_error(&mysql)[0] != '\0')
{
    // une erreur est survenue
}
```

La langue des messages d'erreurs peut être changèe en recompilant la librairie du client MySQL. Actuellement, vous pouvez choisir les messages d'erreur parmi un choix de plusieurs langues. Voir Section 4.6.2 [Languages], page 287.

Valeur de retour

Une chaîne de caractéres qui dècrit l'erreur. Une chaîne vide si aucune erreur n'est survenue.

Erreurs

Aucune.

8.4.3.14 mysql_escape_string()

Vous devez utiliser la fonction `mysql_real_escape_string()` á la place de celle ci !

Cette fonction est identique á `mysql_real_escape_string()` á l'exception faite que `mysql_real_escape_string()` prends deux identifiants de connexion comme premiers arguments et èchappe la chaîne en se basant que le jeu de caractéres courant. `mysql_escape_string()` ne prends pas d'identifiant de connexion et ne respecte pas le jeu de caractéres courant.

8.4.3.15 mysql_fetch_field()

```
MYSQL_FIELD *mysql_fetch_field(MYSQL_RES *result)
```

Description

Retourne la dèfinition d'une colonne d'un jeu de rèsultats en tant que structure `MYSQL_FIELD`. Appelez cette fonction plusieurs fois pour obtenir des informations á propos de toutes les colonnes dans le jeu de rèsultat. `mysql_fetch_field()` retourne `NULL` quand il ne reste plus de champs.

`mysql_fetch_field()` est mis á zèro pour retourner des informations á propos du premier champ á chaque fois que vous exècutez une nouvelle requête `SELECT`. Le champ retournè par `mysql_fetch_field()` est aussi affectè par les appels á `mysql_field_seek()`.

Si vous avez appelè `mysql_query()` pour exècuter un `SELECT` sur une table mais n'avez pas appelè `mysql_store_result()`, MySQL retourne la longueur par dèfaut du blob (8K octets) si vous avez appelè `mysql_fetch_field()` pour obtenir la longueur d'un champ `BLOB`. (La taille 8K est choisie car MySQL ne connait pas la longueur maximale du `BLOB`. Cela devrait être un jour configurable.) Une fois que vous avez rècupèrè le jeu de rèsultats, `field->max_length` contient la longueur de la plus grande valeur de cette colonne dans la requête spècifièe.

Valeur de retour

La structure `MYSQL_FIELD` de la colonne courante. `NULL` s'il ne reste plus de colonnes.

Erreurs

Aucune.

Exemple

```
MYSQL_FIELD *field;

while((field = mysql_fetch_field(result)))
{
    printf("nom du champ : %s\n", field->name);
}
```

8.4.3.16 mysql_fetch_fields()

```
MYSQL_FIELD *mysql_fetch_fields(MYSQL_RES *result)
```

Description

Retourne un tableau de toutes les structures `MYSQL_FIELD` dans un jeu de rèsultats. Chaque structure fournit la dèfinition de champ d'une colonne dans le jeu de rèsultats.

Valeur de retour

Un tableau de structures `MYSQL_FIELD` pour toutes les colonnes dans le jeu de rèsultat.

Erreurs

Aucune.

Exemple

```
unsigned int num_fields;
unsigned int i;
MYSQL_FIELD *fields;

num_fields = mysql_num_fields(result);
fields = mysql_fetch_fields(result);
for(i = 0; i < num_fields; i++)
{
   printf("Le champ %u est %s\n", i, fields[i].name);
}
```

8.4.3.17 mysql_fetch_field_direct()

```
MYSQL_FIELD *mysql_fetch_field_direct(MYSQL_RES *result, unsigned int
fieldnr)
```

Description

Etant donnè un numèro de champ `fieldnr` pour une colonne dans un jeu de rèsultats, cette fonction retourne la dèfinition de ce champ en tant que structure `MYSQL_FIELD`. Vous pouvez utiliser cette fonction pour obtenir la dèfinition d'une colonne choisie arbitrairement. La valeur de `fieldnr` doit varier entre 0 et `mysql_num_fields(result)-1`.

Valeur de retour

La structure `MYSQL_FIELD` pour la colonne spècifièe.

Erreurs

Aucune.

Exemple

```
unsigned int num_fields;
unsigned int i;
MYSQL_FIELD *field;

num_fields = mysql_num_fields(result);
for(i = 0; i < num_fields; i++)
{
    field = mysql_fetch_field_direct(result, i);
    printf("La champ %u est %s\n", i, field->name);
}
```

8.4.3.18 `mysql_fetch_lengths()`

`unsigned long *mysql_fetch_lengths(MYSQL_RES *result)`

Description

Retourne les longueurs des colonnes de la ligne courante dans le jeu de rèsultats. Si vous voulez copier les valeurs des champs, cette information sur la longueur est trés utile pour l'optimisation, car vous pouvez èviter les appels á `strlen()`. De plus, si le jeu de rèsultat contient des donnèes binaires, vous **devez** cette fonction pour dèterminer la longueur des donnèes, car `strlen()` retourne des rèsultats incorrects pour les champs contenant des caractéres nuls.
La longueur des colonnes vides et des colonnes contenant la valeur `NULL` est zèro. Pour savoir comment distinguer ces cas, voyez la description de `mysql_fetch_row()`.

Valeur de retour

Un tableau d'entiers longs non-signès reprèsentant la taille de chaque colonne (n'incluant pas la caractére nul de terminaison). `NULL` si une erreur se produit.

Erreurs

`mysql_fetch_lengths()` n'st valide que pour la ligne courante du jeu de rèsultats. Cette fonction retourne `NULL` si vous l'appelez avant d'appeler `mysql_fetch_row()` ou aprés avoir rècupèrè toutes les lignes du rèsultat.

Exemple

```
MYSQL_ROW row;
unsigned long *lengths;
unsigned int num_fields;
unsigned int i;

row = mysql_fetch_row(result);
if (row)
{
    num_fields = mysql_num_fields(result);
    lengths = mysql_fetch_lengths(result);
    for(i = 0; i < num_fields; i++)
    {
         printf("La colonne %u a %l octets de longueur.\n", i, lengths[i]);
    }
}
```

8.4.3.19 `mysql_fetch_row()`

`MYSQL_ROW mysql_fetch_row(MYSQL_RES *result)`

Description

Rècupére la ligne suivante d'un jeu de rèsultats. Lorsqu'elle est utilisèe aprés `mysql_store_result()`, `mysql_fetch_row()` retourne `NULL` quand il n'y a plus de lignes a rècupèrer. Lorsqu'elle est utilisèe aprés `mysql_use_result()`, `mysql_fetch_row()` retourne `NULL` quand il n'y a plus de lignes a rècupèrer ou qu'une erreur est rencontrèe.

Le nombre de valeurs dans la ligne est donnè par `mysql_num_fields(result)`. Si `row` contient la valeur de retour d'un appel á `mysql_fetch_row()`, les pointeurs sur les valeurs sont accèdèes de `row[0]` á `row[mysql_num_fields(result)-1]`. Les valeurs `NULL` de la ligne sont indiquèes par des pointeurs `NULL`.

La longueur de la valeur du champ dans la ligne peut être obtenue en appelant `mysql_fetch_lengths()`. Les champs vides et les champs contenant `NULL` ont tous deux une longueur ègale á zèro; vous pouvez les distinguer en vèrifiant le pointeur sur la valeur du champ. Si le pointeur est `NULL`, le champ est `NULL`; sinon, le champ est vide.

Valeur de retour

Une structure `MYSQL_ROW` pour la prochaine ligne. `NULL` s'il n'y a plus de lignes a rècupèrer ou qu'une erreur survient.

Erreurs

`CR_SERVER_LOST`
: La connexion au serveur a ètè perdue durant la requête.

`CR_UNKNOWN_ERROR`
: Une erreur inconnue est survenue.

Exemple

```
MYSQL_ROW row;
unsigned int num_fields;
unsigned int i;

num_fields = mysql_num_fields(result);
while ((row = mysql_fetch_row(result)))
{
   unsigned long *lengths;
   lengths = mysql_fetch_lengths(result);
   for(i = 0; i < num_fields; i++)
   {
       printf("[%.*s] ", (int) lengths[i], row[i] ? row[i] : "NULL");
   }
   printf("\n");
}
```

8.4.3.20 `mysql_field_count()`

`unsigned int mysql_field_count(MYSQL *mysql)`
Si vous utilisez une version de MySQL plus ancienne que la 3.22.24, vous devez utiliser `unsigned int mysql_num_fields(MYSQL *mysql)`.

Description

Retourne le nombre de colonnes pour la requête la plus rècente de la connexion.

L'utilisation normale de cette fonction est lorsque `mysql_store_result()` a retournè `NULL` (et que vous n'avez donc pas de pointeur sur jeu de rèsultats). Dans ce cas, vous pouvez appeler `mysql_field_count()` pour dèterminer si `mysql_store_result()` aurait dû produire un rèsultat non-vide. Cela permet au programme client d'entreprendre les bonnes actions sans savoir si la requête ètait un `SELECT` (ou èquivalent). L'exemple suivant illustre comment cela peut être fait.

Voir Section 8.4.6.1 [`NULL mysql_store_result()`], page 658.

Valeur de retour

Un entier non-signè reprèsentant le nombre de champs dans un jeu de rèsultats.

Erreurs

Aucune.

Exemple

```
MYSQL_RES *result;
unsigned int num_fields;
unsigned int num_rows;

if (mysql_query(&mysql,query_string))
{
    // erreur
}
else // requête bonne, traitons les données qu'elle renvoit
{
    result = mysql_store_result(&mysql);
    if (result)  // il y'a des lignes
    {
        num_fields = mysql_num_fields(result);
        // rècupére les lignes, puis apelle mysql_free_result(result)
    }
    else  // mysql_store_result() n'a rien retournè; est-ce normal ?
    {
        if(mysql_field_count(&mysql) == 0)
        {
            // la requête ne retourne aucune donnèe
            // (ce n'ètait pas un SELECT)
            num_rows = mysql_affected_rows(&mysql);
        }
        else // mysql_store_result() aurait du retourner des donnèes
        {
            fprintf(stderr, "Erreur : %s\n", mysql_error(&mysql));
        }
    }
}
```

Une alternative est de remplacer l'appel á `mysql_field_count(&mysql)` par `mysql_errno(&mysql)`. Dans ce cas, vous vèrifiez directement les erreurs á partir de `mysql_store_result()` plutôt qu'á partir de `mysql_field_count()` si la requête ètait un `SELECT`.

8.4.3.21 mysql_field_seek()

```
MYSQL_FIELD_OFFSET mysql_field_seek(MYSQL_RES *result, MYSQL_FIELD_OFFSET
offset)
```

Description

Place le pointeur de champs á la position donnèe. Le prochain appel á `mysql_fetch_field()` rècupèrera la dèfinition du champ de la colonne associèe á cet index.
Pour vous placer au dèbut d'une ligne, passez 0 comme valeur d'`offset`.

Valeur de retour

La derniére valeur de l'index de champ.

Erreurs

Aucune.

8.4.3.22 mysql_field_tell()

```
MYSQL_FIELD_OFFSET mysql_field_tell(MYSQL_RES *result)
```

Description

Retourne la position du curseur de champ utilisè pour le dernier appel á `mysql_fetch_field()`. Cette valeur peut être utilisèe en argument de `mysql_field_seek()`.

Valeur de retour

L'indice courant du curseur de champ.

Erreurs

Aucune.

8.4.3.23 mysql_free_result()

```
void mysql_free_result(MYSQL_RES *result)
```

Description

Libére la mèmoire allouè á un rèsultat avec `mysql_store_result()`, `mysql_use_result()`, `mysql_list_dbs()`, etc. Quand vous n'avez plus besoin d'un jeu de rèsultat, vous devez libèrer la mèmoire qu'il occupe en appelant `mysql_free_result()`.

Valeur de retour

Aucune.

Erreurs

Aucune.

8.4.3.24 mysql_get_client_info()

```
char *mysql_get_client_info(void)
```

Description

Retourne une chaîne reprèsentant la version de la librairie du client.

Valeur de retour

Une chaîne de caractéres reprèsentant la version de la librairie du client.

Erreurs

Aucune.

8.4.3.25 mysql_get_host_info()

```
char *mysql_get_host_info(MYSQL *mysql)
```

Description

Retourne une chaîne de caractéres dècrivant le type de connexion actuellement utilisè, incluant le nom du serveur.

Valeur de retour

Une chaîne de caractéres reprèsentant le nom du serveur et le type de connexion.

Erreurs

Aucune.

8.4.3.26 mysql_get_proto_info()

```
unsigned int mysql_get_proto_info(MYSQL *mysql)
```

Description

Retourne la version du protocole utilisè par la connexion courante.

Valeur de retour

Un entier non signè reprèsentant la version du protocole utilisè par la connexion courante.

Erreurs

Aucune.

8.4.3.27 mysql_get_server_info()

```
char *mysql_get_server_info(MYSQL *mysql)
```

Description

Retourne une chaîne reprèsentant le numèro de version du serveur.

Valeur de retour

Une chaîne de caractéres reprèsentant le numèro de version du serveur.

Erreurs

Aucune.

8.4.3.28 mysql_info()

```
char *mysql_info(MYSQL *mysql)
```

Description

Rècupére une chaîne de caractéres fournissant des informations á propos de la requête exècutèe le plus rècemment, mais seulement pour celles listèes ici. Pour les autres requêtes, `mysql_info()` retournera `NULL`. Le format de la chaîne varie selon le type de requête, comme dècrit ici. Les nombres prèsentès sont des exemples; la chaîne retournèe contiendra les informations correspondantes á vos requêtes.

`INSERT INTO ... SELECT ...`
String format: `Records: 100 Duplicates: 0 Warnings: 0`

`INSERT INTO ... VALUES (...),(...),(...)...`
String format: `Records: 3 Duplicates: 0 Warnings: 0`

`LOAD DATA INFILE ...`
String format: `Records: 1 Deleted: 0 Skipped: 0 Warnings: 0`

`ALTER TABLE`
String format: `Records: 3 Duplicates: 0 Warnings: 0`

`UPDATE` String format: `Rows matched: 40 Changed: 40 Warnings: 0`

Notez que `mysql_info()` retourne une valeur non-nulle (`NULL`) pour les requêtes `INSERT ... VALUES` seulement si une liste de valeurs multiples est fournie á la requête.

Valeur de retour

Une chaîne de caractéres reprèsentant des informations additionnelles á propos de la derniére requête exècutèe. `NULL` si aucune information n'est disponible pour la requête.

Erreurs

Aucune.

8.4.3.29 `mysql_init()`

```
MYSQL *mysql_init(MYSQL *mysql)
```

Description

Alloue ou initialise un objet `MYSQL` convenable pour `mysql_real_connect()`. Si `mysql` est un pointeur `NULL`, la fonction alloue, initialise et retourne un nouvel objet. Sinon, l'objet est initialisè et son adresse est retournèe. Si `mysql_init()` alloue un nouvel objet, il sera libèrè quand `mysql_close()` sera appelèe pour clore la connexion.

Valeur de retour

Un gestionnaire `MYSQL*` initialisè. `NULL` s'il n'y avait pas assez de mèmoire pour allouer le nouvel objet.

Erreurs

Si la mèmoire est insuffisante, `NULL` est retournè.

8.4.3.30 `mysql_insert_id()`

```
my_ulonglong mysql_insert_id(MYSQL *mysql)
```

Description

Retourne l'identifiant gènèrè pour une colonne `AUTO_INCREMENT` par la derniére requête. Utilisez cette commande aprés avoir exècutè une requête `INSERT` sur une table qui contient un champ `AUTO_INCREMENT`.

Notez que `mysql_insert_id()` retourne `0` si la derniére requête n'a pas gènèrè de valeur `AUTO_INCREMENT`. Si vous voulez garder cette valeur pour plus tard, assurez vous d'appeler `mysql_insert_id()` immèdiatement aprés la requête ayant gènèrè cette valeur.

`mysql_insert_id()` est mis á jour aprés l'exècution de requêtes `INSERT` et `UPDATE` qui gènèrent une valeur `AUTO_INCREMENT` ou qui definissent la valeur d'une colonne á `LAST_INSERT_ID(expr)`. Voir Section 6.3.6.2 [Miscellaneous functions], page 472.

Notez aussi que la valeur de retour de la fonction SQL `LAST_INSERT_ID()` contient toujours la valeur d'`AUTO_INCREMENT` la plus á jour. Cette valeur n'est pas remise á zèro lors de l'exècution d'autre requêtes car elle est maintenue pour le serveur.

Valeur de retour

La valeur de la colonne `AUTO_INCREMENT` qui a ètè mise á jour par la derniére requête. Retourne zèro si aucune requête n'avait eu lieu durant la connexion, ou si la derniére requête n'a pas mis á jour la valeur de la colonne `AUTO_INCREMENT`.

Erreurs

Aucune.

8.4.3.31 `mysql_kill()`

```
int mysql_kill(MYSQL *mysql, unsigned long pid)
```

Description

Demande au serveur de terminer le thread spècifiè par `pid`.

Valeur de retour

Zèro si la commande a ètè effectuèe avec succés. Diffèrente de zèro si une erreur est survenue.

Erreurs

`CR_COMMANDS_OUT_OF_SYNC`
: Les commandes n'ont pas ètè exècutèes dans le bon ordre.

`CR_SERVER_GONE_ERROR`
: Le serveur MySQL ne rèponds pas.

`CR_SERVER_LOST`
: La connexion au serveur a ètè perdue au cours la requête.

`CR_UNKNOWN_ERROR`
: Une erreur inconnue s'est produite.

8.4.3.32 `mysql_list_dbs()`

```
MYSQL_RES *mysql_list_dbs(MYSQL *mysql, const char *wild)
```

Description

Retourne une jeu de rèsultats se composant des noms des bases de donnèes localisèes sur le serveur qui correspondent á l'expression règuliére spècifiè par le paramétre `wild`. `wild` peut contenir les caractéres spèciaux '`%`' ou '`_`', ou peut être un pointeur `NULL` pour obtenir la liste de toutes les bases de donnèes. Utiliser `mysql_list_dbs()` reviens á exècuter la requête `SHOW databases [LIKE wild]`.

Vous devez libèrer le rèsultat avec `mysql_free_result()`.

Valeur de retour

Un jeu de rèsultats `MYSQL_RES` en cas de succés. `NULL` si une erreur est survenue.

Erreurs

`CR_COMMANDS_OUT_OF_SYNC`
: Les commandes n'ont pas ètè exècutèes dans le bon ordre.

`CR_SERVER_GONE_ERROR`
: Le serveur MySQL ne rèponds pas.

`CR_SERVER_LOST`
: La connexion au serveur a ètè perdue au cours la requête.

`CR_UNKNOWN_ERROR`
: Une erreur inconnue s'est produite.

8.4.3.33 `mysql_list_fields()`

```
MYSQL_RES *mysql_list_fields(MYSQL *mysql, const char *table, const char
*wild)
```

Description

Retourne un jeu de rèsultats consistant des noms de champs dans une table qui correspondent l'expression règuliére simple spècifièe par la paramétre `wild`. `wild` peut contenir les caractéres spèciaux '`%`' ou '`_`', ou peut être un pointeur `NULL` pour correspondre á tous les

champs. Utiliser `mysql_list_fields()` revient á exècuter la requête `SHOW COLUMNS FROM nom_de_table [LIKE wild]`.

Notez qu'il est recommandè d'utiliser `SHOW COLUMNS FROM nom_de_table` au lieu de /`mysql_list_fields()`.

Vous devez libèrer le rèsultat avec `mysql_free_result()`.

Valeur de retour

Un jeu de rèsultats `MYSQL_RES` en cas de succés. `NULL` sinon.

Erreurs

`CR_COMMANDS_OUT_OF_SYNC`
: Les commandes n'ont pas ètè exècutèes dans le bon ordre.

`CR_SERVER_GONE_ERROR`
: Le serveur MySQL ne rèponds pas.

`CR_SERVER_LOST`
: La connexion au serveur a ètè perdue au cours la requête.

`CR_UNKNOWN_ERROR`
: Une erreur inconnue s'est produite.

8.4.3.34 `mysql_list_processes()`

```
MYSQL_RES *mysql_list_processes(MYSQL *mysql)
```

Description

Retourne un jeu de rèsultat dècrivant les threads courants du serveur. C'est le même genre d'informations renvoyè par `mysqladmin processlist` ou une requête `SHOW PROCESSLIST`.

Vous devez libèrer le jeu de rèsultat avec `mysql_free_result()`.

Valeur de retour

Un jeu de rèsultat `MYSQL_RES` en cas de succés. `NULL` si une erreur est survenue.

Erreurs

`CR_COMMANDS_OUT_OF_SYNC`
: Les commandes n'ont pas ètè exècutèes dans le bon ordre.

`CR_SERVER_GONE_ERROR`
: Le serveur MySQL ne rèponds pas.

`CR_SERVER_LOST`
: La connexion au serveur a ètè perdue au cours la requête.

`CR_UNKNOWN_ERROR`
: Une erreur inconnue s'est produite.

8.4.3.35 `mysql_list_tables()`

```
MYSQL_RES *mysql_list_tables(MYSQL *mysql, const char *wild)
```

Description

Retourne un jeu de rèsultats consistant des noms de tables dans la base de donnèes courante qui concordent avec l'expression règuliére spècifiè par le paramétre `wild`. `wild` peut contenir les caractéres spèciaux '`%`' ou '`_`', ou peut être un pointeur `NULL` pour obtenir toutes les tables. Faire appel á `mysql_list_tables()` revient á exècuter la requête `SHOW tables [LIKE wild]`.

Vous devez libèrer le jeu de rèsultats avec `mysql_free_result()`.

Valeur de retour

Un jeu d'enregistrements `MYSQL_RES` en cas de succés. `NULL` en cas d'erreurs.

Erreurs

`CR_COMMANDS_OUT_OF_SYNC`
: Les commandes n'ont pas ètè exècutèes dans le bon ordre.

`CR_SERVER_GONE_ERROR`
: Le serveur MySQL ne rèponds pas.

`CR_SERVER_LOST`
: La connexion au serveur a ètè perdue au cours la requête.

`CR_UNKNOWN_ERROR`
: Une erreur inconnue s'est produite.

8.4.3.36 `mysql_num_fields()`

```
unsigned int mysql_num_fields(MYSQL_RES *result)
```

ou

```
unsigned int mysql_num_fields(MYSQL *mysql)
```

La seconde forme ne fonctionne plus á partir de la version 3.22.24 de MySQL. Pour passer un argument `MYSQL*`, vous devez utiliser la fonction `unsigned int mysql_field_count(MYSQL *mysql)` á la place.

Description

Retourne le nombre de colonnes dans un jeu de rèsultats.

Notez que vous pouvez obtenir le nombre de colonnes soit á partir d'un pointeur sur rèsultat, soit d'un pointeur de connexion. Vous utiliserez le pointeur de connexion si `mysql_store_result()` ou `mysql_use_result()` ont retournès `NULL` (et que donc, vous n'avez pas de pointeur sur rèsultat). Dans ce cas, vous pouvez appeler `mysql_field_count()` pour dèterminer si `mysql_store_result()` aurait du retourner un rèsultat non-vide. Cela permet au client d'effectuer les bonnes actions sans savoir si la requête ètait un `SELECT` (ou èquivalent). L'exemple ci-dessous montre comment cela doit être utilisè.

Voir Section 8.4.6.1 [`NULL mysql_store_result()`], page 658.

Valeur de retour

Un entier non-signè reprèsentant le nombre de champs dans un jeu de rèsultats.

Erreurs

Aucune.

Exemple

```
MYSQL_RES *result;
unsigned int num_fields;
unsigned int num_rows;

if (mysql_query(&mysql,query_string))
{
    // erreur
}
else // la requête fonctionne, on s'occupe des données
{
    result = mysql_store_result(&mysql);
    if (result)  // il y'a des lignes
    {
        num_fields = mysql_num_fields(result);
        // recupèrer les lignes, puis appeler mysql_free_result(result)
    }
    else  // mysql_store_result() n'a rien retournè ! pourquoi ?
    {
        if (mysql_errno(&mysql))
{
           fprintf(stderr, "Erreur : %s\n", mysql_error(&mysql));
}
        else if (mysql_field_count(&mysql) == 0)
        {
```

```
                // la requête ne retourne pas de données
                // (ce n'etait pas un SELECT)
                num_rows = mysql_affected_rows(&mysql);
            }
        }
    }
```

Une alternative (si vous savez que votre requête aurait du retourner des rèsultats) est de remplacer l'appel á `mysql_errno(&mysql)` par un test sur la nullitè de `mysql_field_count(&mysql)`. Cela n'arrive que si un probléme a ètè rencontrè.

8.4.3.37 mysql_num_rows()

```
my_ulonglong mysql_num_rows(MYSQL_RES *result)
```

Description

Retourne le nombre de lignes prèsentes dans le rèsultat.

L'utilisation de `mysql_num_rows()` dèpend de si vous utilisez `mysql_store_result()` ou `mysql_use_result()` pour retourner le jeu rèsultat. Si vous utilisez `mysql_store_result()`, `mysql_num_rows()` peut être appelè immèdiatement. Si vous utilisez `mysql_use_result()`, `mysql_num_rows()` ne retournera pas la valeur correcte tant que toutes les lignes du rèsultat n'auront pas ètè rècupèrèes.

Valeur de retour

Le nombre de lignes dans le rèsultat.

Erreurs

Aucune.

8.4.3.38 mysql_options()

```
int mysql_options(MYSQL *mysql, enum mysql_option option, const char *arg)
```

Description

Cette fonction peut être utilisèe pour spècifier des options de connexion et modifier le comportement de la session courante. Cette fonction peut être appelèe plusieurs fois pour dèfinir plusieurs options.

`mysql_options()` doit être appelèe aprés `mysql_init()` et avant `mysql_connect()` ou `mysql_real_connect()`.

L'argument `option` est l'option que vous voulez configurer; l'arguement `arg` est la valeur pour cette option. Si l'option est un entier, `arg` doit pointer sur la valeur d'un entier.

Les valeurs possibles pour les options sont :

Option	Type de l'argument	Fonction
MYSQL_OPT_CONNECT_TIMEOUT	unsigned int *	Dèlai d'inactivitè maximal permis.
MYSQL_OPT_COMPRESS	Non utilisè	Utiliser le protocole compressè client/serveur.
MYSQL_OPT_LOCAL_INFILE	pointeur optionnel sur uint	Si aucun pointeur n'est donnè, ou que celui-ci pointe sur un unsigned int != 0 la commande LOAD LOCAL INFILE est activèe.
MYSQL_OPT_NAMED_PIPE	Non utilisè	Utiliser les tunnels nommès pour se connecter au serveur MySQL sur NT.
MYSQL_INIT_COMMAND	char *	Commande á exècuter lors de la connexion au serveur MySQL. Sera automatiquement rè-exècutèe lors des reconnexions.
MYSQL_READ_DEFAULT_FILE	char *	Lit les options á partir du fichier d'options nommè plutôt que de 'my.cnf'.
MYSQL_READ_DEFAULT_GROUP	char *	Lit les options á partir du groupe spècifiè dans le fichier d'options 'my.cnf'ou le fichier spècifiè par MYSQL_READ_DEFAULT_FILE.

Notez que le groupe client est toujours lu si vous utilisez MYSQL_READ_DEFAULT_FILE ou MYSQL_READ_DEFAULT_GROUP.

Le groupe spècifiè dans le fichier des options peut contenir les options suivantes :

Option	Description
connect-timeout	Dèlai d'inactivitè maximal permis en secondes. Sous Linux, ce dèlai est aussi utilisè lors de l'attente de la premiére rèponse du serveur.
compress	Utiliser le protocole compressè client/serveur.
database	Se connecter á cette base de donnèes si aucune base n'a ètè sèlectionnèe á la connexion.
debug	Options de debogage.
disable-local-infile	Interdit l'utilisation de LOAD DATA LOCAL.
host	Nom d'hôte par dèfaut.
init-command	Commande lors de la connexion au serveur MySQL. Sera automatiquement rè-exècutèe lors des reconnexions.
interactive-timeout	Revient á spècifier CLIENT_INTERACTIVE á mysql_real_connect(). Voir Section 8.4.3.41 [mysql_real_connect], page 645.

`local-infile[=(0|1)]`	Si aucun argument, ou argument != 0, on permet alors l'utilisation de `LOAD DATA LOCAL`.
`password`	Mot de passe par dèfaut.
`pipe`	Utiliser les tunnels nommès pour se connecter á MySQL sur NT.
`port`	Port par dèfaut.
`return-found-rows`	Demande á `mysql_info()` de retourner les lignes trouvèes au lieu des lignes mises á jour lors de l'utilisation de `UPDATE`.
`socket`	Numèro de socket par dèfaut.
`user`	Utilisateur par dèfaut.

Notez que `timeout` a ètè remplacè par `connect-timeout`, mais que `timeout` fonctionne encore pour le moement.

Pour plus d'informations sur les fichiers d'options, reportez vous á Section 4.1.2 [Option files], page 198.

Valeur de retour

Zèro si la commande a ètè effectuèe avec succés. Diffèrente de zèro si une erreur est survenue.

Exemple

```
MYSQL mysql;

mysql_init(&mysql);
mysql_options(&mysql,MYSQL_OPT_COMPRESS,0);
mysql_options(&mysql,MYSQL_READ_DEFAULT_GROUP,"odbc");
if (!mysql_real_connect(&mysql,"host","user","passwd","database",0,NULL,0))
{
    fprintf(stderr, "Impossible de se connecter á la base de données. Erreur : %s\n
          mysql_error(&mysql));
}
```

Ce qui prècéde demande au client d'utiliser le protocole compressè client/serveur et lit les options optionnelles de la section `odbc` dans le fichier '`my.cnf`'.

8.4.3.39 `mysql_ping()`

```
int mysql_ping(MYSQL *mysql)
```

Description

Vèrifie si la connexion au serveur est encore assurèe. Si ce n'est pas le cas, une re-connexion automatique est tentèe.

Cette fonction peut être utilisèe par les clients qui restent inactifs longtemps, pour vèrifier que le serveur n'a pas fermè la connexion et se re-connecter si nècessaire.

Valeur de retour

Zèro si le serveur rèpond. Autre que zèro si une erreur est survenue.

Erreurs

`CR_COMMANDS_OUT_OF_SYNC`
: Les commandes n'ont pas ètè exècutèes dans le bon ordre.

`CR_SERVER_GONE_ERROR`
: Le serveur MySQL ne rèponds pas.

`CR_UNKNOWN_ERROR`
: Une erreur inconnue s'est produite.

8.4.3.40 `mysql_query()`

```
int mysql_query(MYSQL *mysql, const char *query)
```

Description

Exècute la requête SQL pointèe par la chaîne terminèe par null `query`. La requête doit se composer d'une seule opèration. Vous ne devez pas ajouter de caractére de terminaison (‘`;`’) ou `\g` á la fin de la requête.

`mysql_query()` ne peut être utilisèe pour les requêtes contenant des donnèes bianaires, vous devez utiliser `mysql_real_query()` á la place. (LEs donnèes binaires peuvent contenir le caractére ‘`\0`’, qui est interprètè comme la fin de la chaîne requête.)

Si vous voulez savoir si la requête doit retourner un jeu de rèsultat ou non, vous pouvez utiliser `mysql_field_count()` pour vèrifier. Voir Section 8.4.3.20 [`mysql_field_count()`], page 630.

Valeur de retour

Zèro si la requête a ètè effectuèe avec succés. Diffèrente de zèro si une erreur est survenue.

Erreurs

`CR_COMMANDS_OUT_OF_SYNC`
: Les commandes n'ont pas ètè exècutèes dans le bon ordre.

`CR_SERVER_GONE_ERROR`
: Le serveur MySQL ne rèponds pas.

`CR_SERVER_LOST`
: La connexion au serveur a ètè perdue au cours la requête.

`CR_UNKNOWN_ERROR`
: Une erreur inconnue s'est produite.

8.4.3.41 mysql_real_connect()

```
MYSQL *mysql_real_connect(MYSQL *mysql, const char *host, const char *user,
const char *passwd, const char *db, unsigned int port, const char *unix_socket,
unsigned int client_flag)
```

Description

`mysql_real_connect()` essaye de se connecter á une base de donnèes MySQL tournant sur l'hôte. `mysql_real_connect()` doit se terminer correctement avant que vous ne puissiez aucune autre fonction de l'API, á l'exception de `mysql_get_client_info()`.

Les paramétres sont spècifiès comme suit :

- Le premier paramétre doit être l'adresse d'une structure `MYSQL` existante. Avant d'appeler `mysql_real_connect()` vous devez appeler `mysql_init()` pour initialiser la structure `MYSQL`. Vous pouvez changer un tas d'options de connexion en appelant `mysql_options()`. Voir Section 8.4.3.38 [mysql_options], page 641.
- La valeur de `host` peut être un nom de domaine ou une adresse IP. Si `host` est `NULL` ou ègal á la chaîne `"localhost"`, une connexion á la machine local est essayèe. Si le systéme supporte les sockets (Unix) ou les tunnels nommès (Windows), elles sont utilisèes au lieu de TCP/IP pour se connecter au serveur.
- La paramétre `user` contient l'identifiant MySQL de l'utilisateur. Si `user` est `NULL`, l'utilisateur courant est sous-entendu. Avec Unix c'est l'utilisateur courant. Avec Windows ODBC, le nom de l'utilisateur courant doit être spècifiè explicitement. Voir Section 8.3.2 [ODBC administrator], page 600.
- La paramétre `passwd` contient le mot de passe de `user`. Si `passwd` est `NULL`, seules les entrèes de la table `user` pour l'utilisateur ayant un champ vide seront testèes. Cela permet á l'administrateur de mettre en place le systéme de priviléges MySQL de façon á ce que les utilisateurs aient divers priviléges selon qu'ils aient spècifiè ou pas de mot de passe.

  Note : N'essayez pas d'encrypter le mot de passe avant l'appel á `mysql_real_connect()`; l'encryptage du mot de passe est gèrèe automatiquement par l'API du client.
- `db` est le nom de la base de donnèes. Si `db` n'est pas `NULL`, la connexion changera la base par dèfaut en cette valeur.
- Si `port` est diffèrent de 0, sa valeur sera utilisè comme port de connexion TCP/IP. Notez que le paramétre `host` dètermine le type de la connexion.
- Si `unix_socket` n'est pas `NULL`, la chaîne spècifie la socket ou le tunnel nommè á utiliser. Notez que le paramétre `host` dètermine le type de la connexion.
- La valeur de `client_flag` est habituellement 0, mais peut être la combinaison des options suivantes dans des circonstances trés spèciales :

Nom de l'option	Description
`CLIENT_COMPRESS`	Utilise le protocole compressè.
`CLIENT_FOUND_ROWS`	Retourne le nombre de lignes trouvèes, et non de lignes affectèes.

`CLIENT_IGNORE_SPACE`	Autorise les espaces aprés les noms de fonctions. Rend tous les noms de fonctions des mots rèservès.
`CLIENT_INTERACTIVE`	Autorise `interactive_timeout` secondes (au lieu de `wait_timeout` secondes) d'innactivitè avant de fermer la connexion.
`CLIENT_NO_SCHEMA`	N'autorise pas la syntaxe `db_name.tbl_name.col_name`. Cela est fait pour ODBC. Il fait gènérer une erreur á l'analyseur si vous utilisez cette syntaxe, ce qui peut se rèveler fort utile pour la chasse aux bogues dans les programmes ODBC.
`CLIENT_ODBC`	Le client est un client ODBC. Cela rend `mysqld` plus accueillant vis á vis de ODBC.
`CLIENT_SSL`	Utilise SSL (protocole encryptè).

Valeur de retour

Un gestionnaire de connexion `MYSQL*` si la connexion a rèussi, `NULL` si elle a èchouè. Pour une connexion á succés, la valeur de retour est la même que la valeur du premier paramétre.

Erreurs

`CR_CONN_HOST_ERROR`
: Impossible de se connecter au serveur MySQL.

`CR_CONNECTION_ERROR`
: Impossible de se connecter au serveur MySQL local.

`CR_IPSOCK_ERROR`
: Impossible de crèer une socket IP.

`CR_OUT_OF_MEMORY`
: Plus de mèmoire.

`CR_SOCKET_CREATE_ERROR`
: Impossible de crèer une socket UNIX.

`CR_UNKNOWN_HOST`
: Impossible de trouver l'adresse IP de l'hôte.

`CR_VERSION_ERROR`
: Une disparitè de protocole a rèsultè de la tentative de connexion á un serveur avec une librairie de client qui utilise une version diffèrente du protocole. Cela peut arriver si vous utilisez une trés vieille librairie cliente pour vous connecter á un serveur qui n'a pas ètè dèmarrè avec l'option `--old-protocol`.

`CR_NAMEDPIPEOPEN_ERROR`
: Impossible de crèer un tunnel nommè sur Windows.

`CR_NAMEDPIPEWAIT_ERROR`
: Impossible d'attendre un tunnel nommè sur Windows.

`CR_NAMEDPIPESETSTATE_ERROR`
: Impossible d'obtenir un gestionnaire de tunnel sur Windows.

`CR_SERVER_LOST`
: Si `connect_timeout` > 0 et qu'il a fallu plus de `connect_timeout` secondes pour se connecter au serveur, ou que celui-ci n'a plus rèpondu durant l'exècution de `init-command`.

Exemple

```
MYSQL mysql;

mysql_init(&mysql);
mysql_options(&mysql,MYSQL_READ_DEFAULT_GROUP,"your_prog_name");
if (!mysql_real_connect(&mysql,"host","user","passwd","database",0,NULL,0))
{
    fprintf(stderr, "Impossible de se connecter á la base de données, erreur : %s\n
          mysql_error(&mysql));
}
```

En utilisant `mysql_options()` la librairie MySQL lira les sections `[client]` et `[your_prog_name]` dans le fichier '`my.cnf`' ce qui assurera le bon fonctionnement de votre programme, même si quelqu'un a configurè MySQL d'une façon non-standard.

Notez que pendant la connexion, `mysql_real_connect()` configure l'option `reconnect` (partie de la structure `MYSQL`) á `1`. Cette option indique, dans le cas oú une requête ne peut être exècutèe á cause d'une dèconnexion, d'essayer de se reconnecter au serveur avant d'abandonner.

8.4.3.42 `mysql_real_escape_string()`

```
unsigned long mysql_real_escape_string(MYSQL *mysql, char *en, const char *de,
unsigned long longueur)
```

Description

Cette fonction est utilisèe pour crèer une requête SQL lègale que vous pouvez utiliser dans une commande SQL. Voir Section 6.1.1.1 [String syntax], page 404.

La string dans `de` est encodèe en chaîne èchappè SQL, prenom en compte le jeu de caractéres de la connexion. Le rèsultat est placè dans `en` et un octet nul de terminaison est ajoutè á la fin de celui-ci. Les caractéres encodès sont `NUL` (ASCII 0), '`\n`', '`\r`', '`\`', '`'`', '`"`', et Ctrl-Z (voir Section 6.1.1 [Literals], page 404). (En fait, MySQL a seulement besoin que l'anti-slash et le guillemet utilisè pour entourer la chaîne soient èchappès. Cette fonction èchappe les autre caractéres pour les rendre plus facile á lire dans les fichiers de log.)

La chaîne pointèe par `de` doit avoir une taille de `longueur` octets. Vous devez allouer á l'espace de `en` au moins `longueur*2+1` octets. (Dans le pire des cas, chaque caractére devra être encodè en utilisant deux octets, et vous avez besoin de place pour l'octet nul de terminaison.) Lorsque `mysql_escape_string()` retourne un rèsultat, le contenu de `en` sera une chaîne terminèe par un caractére nul. La valeur de retour est la longueur de la chaîne encodèe, n'incluant pas le caractére nul de terminaison.

Exemple

```
char query[1000],*end;

end = strmov(query,"INSERT INTO test_table values(");
*end++ = '\'';
end += mysql_real_escape_string(&mysql, end,"C'est quoi ça",11);
*end++ = '\'';
*end++ = ',';
*end++ = '\'';
end += mysql_real_escape_string(&mysql, end,"donnèe binaire : \0\r\n",16);
*end++ = '\'';
*end++ = ')';

if (mysql_real_query(&mysql,query,(unsigned int) (end - query)))
{
   fprintf(stderr, "Impossible d'insèrer la ligne, erreur : %s\n",
           mysql_error(&mysql));
}
```

La fonction `strmov()` utilisèe dans cet exemple est inclue dans la librairie `mysqlclient` et fonctionne comme `strcpy()` mais retourne un pointeur sur le nul de fin du premier paramétre.

Valeur de retour

La longueur de la valeur passèe dans `to`, n'incluant pas la caractére nul de fin de chaîne.

Erreurs

Aucune.

8.4.3.43 `mysql_real_query()`

```
int mysql_real_query(MYSQL *mysql, const char *query, unsigned long length)
```

Description

Exècute la requête SQL pointèe par `query`, qui doit être une chaîne de caractéres de `length` octets de longueur. La requête ne doit contenir qu'une seule commande. Vous ne devez pas ajouter de point virgule (';') ou `\g` á la fin de la requête.

Vous **devez** utiliser `mysql_real_query()` au lieu de `mysql_query()` pour les requêtes qui continent des données binaires, car celles-ci peuvent contenir le caractére'`\0`'. De plus, `mysql_real_query()` est plus rapide que `mysql_query()` car elle n'invoque pas `strlen()` sur la chaîne contenant la requête.

Si vous voulez savoir si la requête est censèe retourner un jeu de rèsultat ou non, vous pouvez utiliser `mysql_field_count()` pour vèrifier cela. Voir Section 8.4.3.20 [mysql_field_count], page 630.

Valeur de retour

Zèro si la requête a ètè effectuèe avec succés. Diffèrente de zèro si une erreur est survenue.

Erreurs

`CR_COMMANDS_OUT_OF_SYNC`
: Les commandes n'ont pas ètè exècutèes dans le bon ordre.

`CR_SERVER_GONE_ERROR`
: Le serveur MySQL ne rèponds pas.

`CR_SERVER_LOST`
: La connexion au serveur a ètè perdue au cours la requête.

`CR_UNKNOWN_ERROR`
: Une erreur inconnue s'est produite.

8.4.3.44 `mysql_reload()`

```
int mysql_reload(MYSQL *mysql)
```

Description

Demande au serveur MySQL de recharger les tables de droits. L'utilisateur soit avoir les droits `RELOAD`.

Cette fonction est dèconseillèe. Il est prèfèrable d'utiliser `mysql_query()` pour exècuter une requête `FLUSH PRIVILEGES` á la place.

Valeur de retour

Zèro si la commande a ètè effectuèe avec succés. Diffèrente de zèro si une erreur est survenue.

Erreurs

`CR_COMMANDS_OUT_OF_SYNC`
: Les commandes n'ont pas ètè exècutèes dans le bon ordre.

`CR_SERVER_GONE_ERROR`
: Le serveur MySQL ne rèponds pas.

`CR_SERVER_LOST`
: La connexion au serveur a ètè perdue au cours la requête.

`CR_UNKNOWN_ERROR`
: Une erreur inconnue s'est produite.

8.4.3.45 mysql_row_seek()

```
MYSQL_ROW_OFFSET mysql_row_seek(MYSQL_RES *result, MYSQL_ROW_OFFSET offset)
```

Description

Dèplace le curseur de ligne vers une ligne arbitraire dans un jeu de rèsultats de requête. Cela nècessite que le jeu de rèsultats contienne la totalitè des lignes retournèe par la requête, et donc, `mysql_row_seek()` ne peut être utilisèe qu'en conjonction avec `mysql_store_result()`, et non avec `mysql_use_result()`.

La position doit être une valeur retournèe par un appel á `mysql_row_tell()` ou á `mysql_row_seek()`. Cette valeur n'est pas un simple numèro de ligne; si vous voulez vous dèplacer dans un jeu de rèsultats en utilisant le numèro d'une ligne, utilisez `mysql_data_seek()`.

Valeur de retour

La position prècèdente du curseur de ligne. Cette valeur peut être passèe á `mysql_row_seek()`.

Erreurs

Aucune.

8.4.3.46 mysql_row_tell()

```
MYSQL_ROW_OFFSET mysql_row_tell(MYSQL_RES *result)
```

Description

Retourne la position courante du pointeur de lignes pour le dernier appel á `mysql_fetch_row()`. Cette valeur peut être utilisèe comme argument de `mysql_row_seek()`.

Vous ne devez utiliser `mysql_row_tell()` qu'aprés `mysql_store_result()`, et non aprés `mysql_use_result()`.

Valeur de retour

La position courante du pointeur de ligne.

Erreurs

Aucune.

8.4.3.47 mysql_select_db()

```
int mysql_select_db(MYSQL *mysql, const char *db)
```

Description

Rend la base de donnèes spècifièe par `db` la base par dèfaut (courante) pour la connexion spècifièe par `mysql`. Pour les requêtes suivantes, cette base de donnèes sera utilisèe comme rèfèrence pour les tables dont la base de donnèes n'a pas ètè spècifièe explicitement.

`mysql_select_db()` èchoue si l'utilisateur ne peut être reconnu ayant droit d'accés á la base de donnèes.

Valeur de retour

Zèro si la commande a ètè effectuèe avec succés. Diffèrente de zèro si une erreur est survenue.

Erreurs

`CR_COMMANDS_OUT_OF_SYNC`
: Les commandes n'ont pas ètè exècutèes dans le bon ordre.

`CR_SERVER_GONE_ERROR`
: Le serveur MySQL ne rèponds pas.

`CR_SERVER_LOST`
: La connexion au serveur a ètè perdue au cours la requête.

`CR_UNKNOWN_ERROR`
: Une erreur inconnue s'est produite.

8.4.3.48 `mysql_shutdown()`

```
int mysql_shutdown(MYSQL *mysql)
```

Description

Demande au serveur de base de donnèes de se terminer. L'utilisateur connectè doit avoir le droit `SHUTDOWN`.

Valeur de retour

Zèro si la commande a ètè effectuèe avec succés. Diffèrente de zèro si une erreur est survenue.

Erreurs

`CR_COMMANDS_OUT_OF_SYNC`
: Les commandes n'ont pas ètè exècutèes dans le bon ordre.

`CR_SERVER_GONE_ERROR`
: Le serveur MySQL ne rèponds pas.

`CR_SERVER_LOST`
La connexion au serveur a ètè perdue au cours la requête.

`CR_UNKNOWN_ERROR`
Une erreur inconnue s'est produite.

8.4.3.49 `mysql_stat()`

```
char *mysql_stat(MYSQL *mysql)
```

Description

Retourne une chaîne de caractéres contenant des informations similaires á celle fournies par la commande `mysqladmin status`. Cela inclus le temps de fonctionnement en secondes et le nombre de threads en cours d'exècution, questions, rechargement, et tables ouvertes.

Valeur de retour

Une chaîne de caractéres dècrivant l'ètat du serveur. `NULL` si une erreur est survenue.

Erreurs

`CR_COMMANDS_OUT_OF_SYNC`
Les commandes n'ont pas ètè exècutèes dans le bon ordre.

`CR_SERVER_GONE_ERROR`
Le serveur MySQL ne rèponds pas.

`CR_SERVER_LOST`
La connexion au serveur a ètè perdue au cours la requête.

`CR_UNKNOWN_ERROR`
Une erreur inconnue s'est produite.

8.4.3.50 `mysql_store_result()`

```
MYSQL_RES *mysql_store_result(MYSQL *mysql)
```

Description

Vous devez appeler `mysql_store_result()` ou `mysql_use_result()` pour chaque requête qui rècupére des donnèes avec succés (`SELECT`, `SHOW`, `DESCRIBE`, `EXPLAIN`).

Vous n'avez pas á appeler `mysql_store_result()` ou `mysql_use_result()` pour d'autres requêtes, mais cela ne posera pas de problémes ou ne ralentira pas vos scripts si vous appelez `mysql_store_result()` en tout cas. Vous pouvez savoir si la requête n'a pas renvoyè de rèsultat en vèrifiant si `mysql_store_result()` retourne 0 (nous verrons cela plus tard).

Si vous voulez savoir si la requête devrait renvoyer un jeu de rèsultats ou non, vous pouvez utiliser `mysql_field_count()` pour vèrifier. Voir Section 8.4.3.20 [mysql_field_count], page 630.

`mysql_store_result()` lit le rèsultat en entier et le stocke dans le client, alloue une structure `MYSQL_RES`, et place le rèsultat dans cette structure.

`mysql_store_result()` retourne un pointeur nul si la requête n'a pas retournè un jeu de rèsultats (si la requête ètait, par exemple, un `INSERT`).

`mysql_store_result()` retourne aussi un pointeur nul si la lecture á partir du jeu de rèsultats èchoue. Vous pouvez vèrifier la prèsence d'erreurs en regardant si `mysql_error()` ne retourne pas de pointeur nul, si `mysql_errno()` retourne `<>` 0, ou si `mysql_field_count()` retourne `<>` 0.

Un jeu de rèsultat vide est retournè si aucune ligne n'est retournèe. (Un jeu de rèsultats vide différe d'un pointeur nul en tant que valeur de retour.)

Une fois que vous avez appelè `mysql_store_result()` et obtenu un rèsultat qui n'est pas un pointeur nul, vous devez appeler `mysql_num_rows()` pour trouver combien de lignes contient le jeu de rèsultats.

Vous pouvez appeler `mysql_fetch_row()` pour rècupérer des lignes á partir du jeu de rèsultats, ou `mysql_row_seek()` et `mysql_row_tell()` pour obtenir ou changer la ligne courante dans le jeu de rèsultats.

Vous devez appeler `mysql_free_result()` une fois que vous n'avez plus besoin du rèsultat.

Voir Section 8.4.6.1 [`NULL mysql_store_result()`], page 658.

Valeur de retour

Une structure de rèsultat `MYSQL_RES`. `NULL` si une erreur survient.

Erreurs

`CR_COMMANDS_OUT_OF_SYNC`
: Les commandes n'ont pas ètè exècutèes dans le bon ordre.

`CR_OUT_OF_MEMORY`
: Plus de mèmoire.

`CR_SERVER_GONE_ERROR`
: Le serveur MySQL ne rèponds pas.

`CR_SERVER_LOST`
: La connexion au serveur a ètè perdue au cours la requête.

`CR_UNKNOWN_ERROR`
: Une erreur inconnue s'est produite.

8.4.3.51 `mysql_thread_id()`

```
unsigned long mysql_thread_id(MYSQL *mysql)
```

Description

Retourne l'identifiant du thread de la connexion courante. Cette valeur peut être utilisèe comme argument de `mysql_kill()` pour terminer ce thread.

Si la connexion est perdue et que vous vous reconnectez via `mysql_ping()`, l'identifiant du thread changera. Cela signifie que cela ne sert á rien de rècupèrer l'identifiant du thread et de le sauvegarder pour l'utiliser plus tard. Vous devez le rècupèrer quand vous en avez besoin.

Valeur de retour

L'identifiant du thread de la connexion courante.

Erreurs

Aucune.

8.4.3.52 `mysql_use_result()`

```
MYSQL_RES *mysql_use_result(MYSQL *mysql)
```

Description

Vous devez appeler `mysql_store_result()` ou `mysql_use_result()` pour chaque requête qui rècupére des donnèes avec succés (`SELECT`, `SHOW`, `DESCRIBE`, `EXPLAIN`).

`mysql_use_result()` initialise un jeu de rèsultats mais ne l'enregistre pas dans le client comme le fait `mysql_store_result()`. A la place, chaque ligne doit être rècupèrè manuellement á l'aide de la commande `mysql_fetch_row()`. Cela lit le rèsultat directement á partir du serveur sans l'enregistrer dans une table temporaire ou un tampon local, ce qui est plus rapide et utilise moins de mèmoire que `mysql_store_result()`. Le client n'allouera de la mèmoire que pour la ligne courante et un tampon de communication qui peut aller jusqu'á `max_allowed_packet` octets.

D'une autre côtè, vous ne devez pas utiliser `mysql_use_result()` si vous faites beaucoup de traitements pour chaque ligne côtè client, ou que le rèsultat est envoyè á un ècran oú l'utilisateur peut entrer `^S` (arrêt dèfilement). Cela bloquera le serveur et empêchera les autres threads de mettre á jour n'importe quelle table á partir de laquelle les donnèes sont lues.

Lors de l'utilisation de `mysql_use_result()`, vous devez exècuter `mysql_fetch_row()` jusqu'á ce que `NULL` soit retournè, sinon, les lignes non retournèe seront inclues dans le jeu de rèsultat de votre prochaine requête. L'API C donnera l'erreur `Commands out of sync; you can't run this command now` si vous oubliez de le faire !

Vous ne devez pas utiliser `mysql_data_seek()`, `mysql_row_seek()`, `mysql_row_tell()`, `mysql_num_rows()`, ou `mysql_affected_rows()` avec un rèsultat retournè par `mysql_use_result()`, de même, vous ne devez pas exècuter d'autres requêtes tant que la commande

`mysql_use_result()` n'est pas terminèe. (Toutefois, aprés avoir rècupèrè toutes les lignes, `mysql_num_rows()` retournera correctement le nombre de lignes rècupèrèes.)

Vous devez appeler `mysql_free_result()` lorsque vous n'avez plus besoin du jeu de rèsultats.

Valeur de retour

Une structure de rèsultat `MYSQL_RES`. `NULL` si une erreur survient.

Erreurs

`CR_COMMANDS_OUT_OF_SYNC`
: Les commandes n'ont pas ètè exècutèes dans le bon ordre.

`CR_OUT_OF_MEMORY`
: Plus de mèmoire.

`CR_SERVER_GONE_ERROR`
: Le serveur MySQL ne rèponds pas.

`CR_SERVER_LOST`
: La connexion au serveur a ètè perdue au cours la requête.

`CR_UNKNOWN_ERROR`
: Une erreur inconnue s'est produite.

8.4.4 Description des fonctions threadèes de C

Vous devez utiliser les fonctions suivantes quand vous voulez crèer un client threadè. Voir Section 8.4.8 [Threaded clients], page 660.

8.4.4.1 `my_init()`

`void my_init(void)`

Description

Cette fonction doit être appelèe une fois dans le programme avant tout appel á une fonction MySQL. Cela initialise quelques variables globales dont MySQL a besoin. Si vous utilisez une librairie client sûr pour les threads, cela appellera aussi `mysql_thread_init()` pour ce thread.

Ceci est automatiquement appelè par `mysql_init()`, `mysql_server_init()` et `mysql_connect()`.

Valeur de retour

Aucune.

8.4.4.2 mysql_thread_init()

```
my_bool mysql_thread_init(void)
```

Description

Cette fonction doit être appelèe á chaque crèation de thread pour initialiser les variables spècifiques aux threads.

Elle est appelèe automatiquement par `my_init()` et `mysql_connect()`.

Valeur de retour

Aucune.

8.4.4.3 mysql_thread_end()

```
void mysql_thread_end(void)
```

Description

Cette fonction doit être appelèe avant `pthread_exit()` pour libèrer la mèmoire allouèe part `mysql_thread_init()`.

Notez que cette fonction **n'est pas invoquèe automatiquement** par la librairie du client. Elle doit être invoquèe explicitement pour èviter les pertes de mèmoire.

Valeur de retour

Aucune.

8.4.4.4 mysql_thread_safe()

```
unsigned int mysql_thread_safe(void)
```

Description

Cette fonction indique si le client est compilè avec le support des threads (thread-safe).

Valeur de retour

1 indique que le client est thread-safe, 0 sinon.

8.4.5 Description des fonctions C du serveur embarquè

Vous devez utiliser les fonctions suivantes si vous voulez permettre á votre application d'être liée avec la librairie du serveur embarquè MySQL. Voir Section 8.4.9 [libmysqld], page 662.

Si le programme est liè avec `-lmysqlclient` au lieu de `-lmysqld`, ces fonctions ne font rien. Cela permet de choisir d'utiliser un serveur embarquè MySQL, ou un serveur tournant á part sans avoir á changer votre code.

8.4.5.1 `mysql_server_init()`

```
int mysql_server_init(int argc, char **argv, char **groups)
```

Description

Cette fonction **doit** être appelèe une fois dans le programme avant d'appeler toute autre fonction MySQL. Elle dèmarre le serveur et initialise tout sous-systéme (`mysys`, InnoDB, etc.) utilisè par le serveur. Si cette fonction n'est pas appelèe, le programme plantera. Si vous utilisez le paquet DBUG fournit avec MySQL, vous devez exècuter cette fonction aprés avoir appelè `MY_INIT()`.

Les arguments `argc` et `argv` sont analogues aux arguments de `main()`. Le premier èlèment `argv` est ignorè (il contient le plus souvent le nom du programme). Par convenance, `argc` peut être `0` (zèro) si il n'y a aucun argument passè en ligne de commande pour le serveur.

La liste de mots terminèe par `NULL` dans `groups` dètermine les groupes dans les fichiers d'options qui seront actifs. Voir Section 4.1.2 [Option files], page 198. Par convenance, `groups` peut être `NULL`, dans ce cas, les groupes `[server]` et `[emedded]` sont activès.

Exemple

```
#include <mysql.h>
#include <stdlib.h>

static char *server_args[] = {
  "ce_programme",       /* cette chaîne n'est pas utilisèe */
  "--datadir=.",
  "--key_buffer_size=32M"
};
static char *server_groups[] = {
  "embedded",
  "server",
  "this_program_SERVER",
  (char *)NULL
};

int main(void) {
  mysql_server_init(sizeof(server_args) / sizeof(char *),
```

```
                        server_args, server_groups);

      /* Utilisez les fonction de L'API MySQL ici */

      mysql_server_end();

      return EXIT_SUCCESS;
    }
```

Valeur de retour

0 en cas de succés, 1 si une erreur survient.

8.4.5.2 `mysql_server_end()`

`void mysql_server_end(void)`

Description

Cette fonction **doit** être appelèe une fois dans le programme aprés toutes les autres fonctions MySQL. Elle coupe le serveur incorporè.

Valeur de retour

Aucune.

8.4.6 Questions courantes sur la librairie C

8.4.6.1 Pourquoi est ce qu'aprés `mysql_query()` ait indiquè un rèsultat positif, `mysql_store_result()` retourne parfois `NULL`?

Il est possible que `mysql_store_result()` retourne `NULL` aprés un appel á `mysql_query()`. Quand cela arrive, cela signifie que l'une des conditions suivantes a ètè remplie :

- Il y'a eu un probléme avec `malloc()` (par exemple, si la taille du rèsultat ètait trop importante).
- Les donnèes n'ont pu être lues (erreur survenue á la connexion).
- La requête n'a retournè aucun rèsultat (par exemple, il s'agissait d'un `INSERT`, `UPDATE`, ou d'un `DELETE`).

Vous pouvez toujours vèrifier si la requête devait bien fournir un rèsultat non vide en invoquant `mysql_field_count()`. Si `mysql_field_count()` retourne zèro, le rèsultat est vide et la derniére requête n'en retournait pas (par exemple, un `INSERT` ou un `DELETE`). Si `mysql_field_count()` retourne un rèsultat non nul, la requête aurait du produire un

rèsultat non nul. Voyez la documentation de la fonction `mysql_field_count()` pour plus d'exemples.

Vous pouvez tester les erreurs en faisant appel á `mysql_error()` ou `mysql_errno()`.

8.4.6.2 Quels rèsultats puis-je obtenir d'une requête?

En plus des enregistrements retournès par une requête, vous pouvez obtenir les informations suivantes :

- `mysql_affected_rows()` retourne le nombre d'enregistrements affectès par la derniére requête `INSERT`, `UPDATE`, ou `DELETE`. Une exception est que si `DELETE` est utilisè sans clause `WHERE`, la table est re-crèèe vide, ce qui est plus rapide! Dans ce cas, `mysql_affected_rows()` retournera zèro comme nombre d'enregistrements affectès.
- `mysql_num_rows()` retourne le nombre d'enregistrements dans le rèsultat. Avec `mysql_store_result()`, `mysql_num_rows()` peut être utilisèe dés que `mysql_store_result()` retourne un rèsultat. Avec `mysql_use_result()`, `mysql_num_rows()` ne doit être appelè qu'aprés avoir rècupèrè tous les enregistrements avec `mysql_fetch_row()`.
- `mysql_insert_id()` retourne l'ID gènèrè par la derniére requête qui a insèrè une ligne dans une table avec un index `AUTO_INCREMENT`. Voir Section 8.4.3.30 [`mysql_insert_id()`], page 635.
- Quelques requêtes (`LOAD DATA INFILE ...`, `INSERT INTO ... SELECT ...`, `UPDATE`) retournent des informations additionnelles. Le rèsultat est renvoyè par `mysql_info()`. Regardez la documentation de `mysql_info()` pour plus d'informations sur le format de la chaîne retournèe. `mysql_info()` retourne un pointeur `NULL` s'il n'y a pas d'informations additionelles.

8.4.6.3 Comment obtenir l'identifiant unique du dernier enregistrement insèrè ?

Si vous insèrez une ligne dans une table qui contient une colonne ayant l'attribut `AUTO_INCREMENT`, vous pouvez obtenir le dernier identifiant gènèrè en appelant la fonction `mysql_insert_id()`.

Vous pouvez aussi rècupèrer cet identifiant en utilisant la fonction `LAST_INSERT_ID()` dans une requête que vous passez á `mysql_query()`.

Vous pouvez vèrifier qu'un index `AUTO_INCREMENT` est utilisè en exècutant le code suivant. Cela vèrifiera aussi si la requête ètait un `INSERT` avec un index `AUTO_INCREMENT` :

```
if (mysql_error(&mysql)[0] == 0 &&
    mysql_num_fields(result) == 0 &&
    mysql_insert_id(&mysql) != 0)
{
    used_id = mysql_insert_id(&mysql);
}
```

L'identifiant gènèrè par la derniére insertion est maintenu sur le serveur en se basant sur la connexion. Il ne sera pas changè par un autre client. Il ne changera pas non plus si vous mettez á jour une autre colonne `AUTO_INCREMENT` avec une valeur normale (ni `NULL` ni `0`).

Si vous voulez utiliser l'identifiant gènèrè pour une table et l'insèrer dans une autre, vous pouvez utiliser les requêtes suivantes :

```
INSERT INTO foo (auto,text)
    VALUES(NULL,'text');              # gènére un identifiant en insèrant NULL
INSERT INTO foo2 (id,text)
    VALUES(LAST_INSERT_ID(),'text');  # on l'utilise dans la seconde page
```

8.4.6.4 Problémes lors de la liaison avec l'API C

Lors de la liaison avec l'API C, l'erreur suivante peut survenir sur quelques systémes :

```
gcc -g -o client test.o -L/usr/local/lib/mysql -lmysqlclient -lsocket -lnsl

Undefined       first referenced
 symbol         in file
floor           /usr/local/lib/mysql/libmysqlclient.a(password.o)
ld: fatal: Symbol referencing errors. No output written to client
```

Si cela se produit sur votre systéme, vous devez inclure la librairie mathèmatique en ajoutant `-lm` á la fin de la ligne de compilation/liaison.

8.4.7 Compiler les clients

Si vous compilez des clients MySQL que vous avez ècrits vous-mêmes, ils doivent être liès en utilisant l'option `-lmysqlclient -lz` de la commande de liaison. Vous aurez peut-être besoin de spècifier l'option `-L` pour dire au programme ou trouver les librairies. Par exemple, si la librairie est installèe dans '`/usr/local/mysql/lib`', utilisez `-L/usr/local/mysql/lib -lmysqlclient -lz` dans votre commande.

Pour les clients qui utilisent les fichiers d'entêtes de MySQL, vous aurez besoin de spècifier une option `-I` lors de leur compilation (par exemple, `-I/usr/local/mysql/include`), pour que le programme puisse les trouver.

8.4.8 Comment faire un client MySQL threadè

La librairie cliente est presque compatible avec les threads. Le probléme le plus important est les routines de '`net.c`' qui lisent les sockets, et qui ne sont pas compatibles avec les interruptions. Cela a ètè fait en imaginant que vous souhaitiez vos propres alarmes, qui pourraient interrompre une lecture trop longue. Si vous installez des gestionnaires d'interruption pour l'alarme `SIGPIPE`, la gestion des sockets devraient être compatible avec les threads.

Dans les anciennes versions binaires que nous distribuions sur notre site web, (`http://www.mysql.com/`), les librairies clientes ètaient normalement compilèes avec l'option de compatibilitè avec les threads (les exècutables Windows sont par dèfaut compatible avec les threads). Les nouvelles distributions binaires doivent disposer des deux librairies, compatibles ou non avec les threads.

Pour obtenir une client threadè oú vous pouvez interrompre le client avec d'autres threads, mettre des dèlais d'expiration lors des discussions avec le seveur MySQL, vous devriez

utiliser les librairies `-lmysys`, `-lmystrings` et `-ldbug`, ainsi que `net_serv.o` que le serveur utilise.

Si vous n'avez pas besoin des interruption ou des expirations, vous pouvez compiler simplement une librairie compatible avec les threads, `(mysqlclient_r)` et l'utiliser. Voir Section 8.4 [MySQL C API], page 609. Dans ce cas, vous n'avez pas á vous prèoccuper du fichier `net_serv.o` ou des autres librairies MySQL.

Lorsque vous utiliser un client threadè et que vous souhaitez utiliser des dèlais d'expiration et des interruptions, vous pouvez faire grand usage des routines du fichier '`thr_alarm.c`'. Si vous utiliser des routines issues de la librairie `mysys`, la seule chose á penser est de commencer par utiliser `my_init()`! Voir Section 8.4.4 [C Thread functions], page 655.

Toutes les fonctions, hormis `mysql_real_connect()` sont compatibles avec les threads par dèfaut. Les notes suivantes dècrivent comment compiler une librairie cliente compatible avec les threads. Les notes ci-dessous, ècrites pour `mysql_real_connect()` s'appliquent aussi á `mysql_connect()`, mais comme `mysql_connect()` est obsoléte, vous devriez utiliser `mysql_real_connect()`).

Pour rendre `mysql_real_connect()` compatible avec les threads, vous devez recompiler la librairie cliente avec cette commande :

```
shell> ./configure --enable-thread-safe-client
```

Cela va crèer une librairie cliente compatible avec les threads `libmysqlclient_r`. Supposons que votre systéme d'exploitation dispose d'une fonction `gethostbyname_r()` compatible avec les threads. Cette librairie est compatible avec les threads pour chaque connexion. Vous pouvez partager une connexion entre deux threads, avec les limitations suivantes :

- Deux threads ne peuvent pas envoyer de requêtes simultanèes au serveur MySQL, sur la même connexion. En particulier, vous devez vous assurer qu'entre `mysql_query()` et `mysql_store_result()`, aucun autre thread n'utilise la même connexion.
- De nombreux threads peuvent accèder á diffèrents rèsultats qui sont lus avec `mysql_store_result()`.
- Si vous utilisez `mysql_use_result`, vous devez vous assurer qu'aucun autre thread n'utilise la même connexion jusqu'á ce qu'elle soit refermèe. Cependant, il vaut bien mieux pour votre client threadè qu'ils utilisent `mysql_store_result()`.
- Si vous voulez utiliser de multiples threads sur la même connexion, vous devez avoir un verrou mutex autour de vos fonctions `mysql_query()` et `mysql_store_result()`. Une fois que `mysql_store_result()` est prêt, le verrou peut être libèrè et d'autres threads vont pouvoir utiliser la connexion.
- Si vous programmez avec les threads POSIX, vous pouvez utiliser les fonctions `pthread_mutex_lock()` et `pthread_mutex_unlock()` pour poser et enlever le verrou mutex.

Vous devez savoir ce qui suit si vous avez un thread qui appel une fonction MySQL qui n,a pas crèe de connexion á la base MySQL :

Lorsque vous appelez `mysql_init()` ou `mysql_connect()`, MySQL va crèer une variable spècifique au thread qui est utilisèe par la libaririe de dèbogage (entre autres).

Si vous appelez une fonction MYSQL, avant que le thread n'ai appelè `mysql_init()` ou `mysql_connect()`, le thread ne va pas avoir les variables spècifiques en place, et vous risquez d'obtenir un core dump tôt ou tard.

Pour faire fonctionner le tout proprement, vous devez suivre ces ètapes :

1. Appeler `my_init()` au dèbut du programme, si il appelle une autre fonction MySQL, avant d'appeler `mysql_real_connect()`.
2. Appeler `mysql_thread_init()` dans le gestionnaire de threads avant d'appeler une autre fonction MySQL.
3. Dans le thread, appelez `mysql_thread_end()` avant d'appeler `pthread_exit()`. Cela va libèrer la mèmoire utiliser par les variables spècifiques MySQL.

Vous pouvez rencontrer des erreurs á cause des symboles non dèfinis lors du link de votre client avec `libmysqlclient_r`. Dans la plupart des cas, c'est parce que vous n'avez pas inclus la librairie de thread dans la ligne de compilation.

8.4.9 libmysqld, la librairie du serveur embarquè MySQL

8.4.9.1 Vue d'ensemble de la librairie du serveur embarquè MySQL

La librairie embarquèe MySQL rend possible l'accés á un serveur MySQL complet, depuis une application. Le principal avantage est l'amèlioration des performances, et une gestion bien plus simple des applications.

Les API sont identiques pour la version embarquèe et la version client/serveur. Pour changer les anciennes applications threadèes, et les faire utiliser la librairie embarquèe, vous devez simplement ajouter deux appels aux fonctions suivantes :

Fonction	**Quand l'utiliser**
`mysql_server_init()`	Doit être appelèe avant toute autre fonction MySQL, et de prèfèrence trés tôt dans la fonction `main()`.
`mysql_server_end()`	Doit être appelèe avant que votre programme ne se termine.
`mysql_thread_init()`	Doit être appelèe dans chaque thread que vous crèez, qui aura accés á MySQL.
`mysql_thread_end()`	Doit être appelèe avant d'appeler `pthread_exit()`

Puis, vous devez compiler votre code avec '`libmysqld.a`' au lieu de '`libmysqlclient.a`'.

Les fonctions ci-dessus `mysql_server_xxx` sont aussi inclues dans la librairie '`libmysqlclient.a`' pour vous permettre de changer facilement entre les versions de la librairie embarquèe et celle de la librairie client/serveur, en compilant simplement la bonne librairie. Voir Section 8.4.5.1 [mysql_server_init], page 657.

8.4.9.2 Compiler des programmes avec `libmysqld`

Pour avoir la librairie `libmysqld` vous devez configurer MySQL avec l'option `--with-embedded-server`.

Quand vous liez votre programme avec `libmysqld`, vous devez aussi inclure les librairies `pthread` spècifiques au systéme et quelques librairies que le serveur MySQL utilise. Vous

pouvez obtenir la liste compléte des librairies en exècutant `mysql_config --libmysqld-libs`.

Les options correctes pour compiler et lier un programme thrèadè doivent être utilisèes, même si vous n'appelez pas directement une fonction de threads dans votre code.

8.4.9.3 Restrictions lors de l'utilisation du serveur embarquè MySQL

Le serveur embarquè posséde les limitations suivantes :

- Pas de support pour les tables ISAM. (Ceci est principalement fait pour rendre la librairie plus petite)
- Pas de fonctions dèfinies par l'utilisateur (UDF).
- Pas de traçage de pile lors des vidages de mèmoire (core dump).
- Pas de support pour les RAID internes. (Cela n'est normalement pas requis vu que la plupart des systémes d'exploitation supportent aujourd'hui les grands fichiers).
- Vous pouvez le configurer en tant que serveur ou maître (pas de replication).
- vous ne pouvez vous connecter au serveur embarquè á partir d'un processus externe avec les sockets ou TCP/IP.

Quelques unes de ces limitations peuvent être changèe en èditant le fichier '`mysql_embed.h`' et recompilant MySQL.

8.4.9.4 Utilisation de fichiers d'options avec le serveur embarquè

Voici la maniére recommandèe d'utiliser les fichiers d'options pour que le passage des applications client/serveur vers une application oú MySQL est embarquè soit plus facile. Voir Section 4.1.2 [Option files], page 198.

- Placez les options communes dans la section `[server]`. Elles seront lues par les deux versions de MySQL.
- Placez les options spècifiques au client/serveur dans la section `[mysqld]`.
- Placez les options spècifiques á MySQL embarquè dans la section `[embedded]`.
- Placez les options spècifiques aux applications dans une section `[NomApplication_SERVER]`.

8.4.9.5 Choses á faire pour le serveur embarquè (TODO)

- Nous ne fournissons pour l'instant qu'une version statique de la librairie `mysqld`, nous fournirons aussi une librairie partagèe pour cela plus tard.
- Nous allons proposer des options pour se dèbarrasser de quelques parties de MySQL pour rendre la librairie plus petite.
- Il y'a encore beaucoup d'optimisations de vitesse á faire.
- Les erreurs sont ècrites dans stderr. Nous ajouterons une options pour spècifier un fichier pour cela.
- Nous devons changer InnoDB pour qu'il n'y ait plus tant de sorties lors de l'utilisation du serveur embarquè.

8.4.9.6 Un exemple simple de serveur embarquè

Ce programme exemple et son makefile devraient fonctionner sans changements sur un systéme Linux ou FreeBSD. Pour les autres systémes d'exploitation, quelques petits changements seront requis. Cet exemple est lá pour vous donner assez de dètails pour comprendre le probléme, sans avoir en tête qu'il s'agit d'une partie nècessaire d'une application rèelle.

Pour essayer ccet exemple, crèez un dossier '`test_libmysqld`' au même niveau que le dossier des sources mysql-4.0. Sauvegardez le fichier source '`test_libmysqld.c`' et '`GNUmakefile`' dans le dossier, puis exècutez GNU '`make`' á l'intèrieur du rèpertoire '`test_libmysqld`'.

'`test_libmysqld.c`'

```
/*
 * Un simple exemple de client, utilisant la librairie du serveur embarquè MySQL
 */

#include <mysql.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>

MYSQL *db_connect(const char *dbname);
void db_disconnect(MYSQL *db);
void db_do_query(MYSQL *db, const char *query);

const char *server_groups[] = {
  "test_libmysqld_SERVER", "embedded", "server", NULL
};

int
main(int argc, char **argv)
{
  MYSQL *one, *two;

  /* mysql_server_init() doit être appelèe avant toute autre fonction
   * mysql.
   *
   * Vous pouvez utiliser mysql_server_init(0, NULL, NULL), et cela initialisera
   * le serveur en utilisant groups = {
   *   "server", "embedded", NULL
   *  }.
   *
   * Dans votre fichier $HOME/.my.cnf, vous voudrez sûrement mettre :

[test_libmysqld_SERVER]
language = /chemin/vers/la/source/de/mysql/sql/share/english

   * Vous pouvez, bien sûr, modifier argc et argv avant de les passer
```

```
   * á cette fonction. Ou vous pouvez en crèer de nouveaux de la maniére
   * que vous souhaitez. Mais tout les arguments dans argv (á part
   * argv[0], qui est le nom du programme) doivent être des options valides
   * pour le serveur MySQL.
   *
   * Si vous liez ce client avec la librairie mysqlclient normale,
   * cette fonction n'est qu'un bout de code qui ne fait rien.
   */
  mysql_server_init(argc, argv, (char **)server_groups);

  one = db_connect("test");
  two = db_connect(NULL);

  db_do_query(one, "SHOW TABLE STATUS");
  db_do_query(two, "SHOW DATABASES");

  mysql_close(two);
  mysql_close(one);

  /* Cela doit être appelè aprés toutes les autres fonctions mysql */
  mysql_server_end();

  exit(EXIT_SUCCESS);
}

static void
die(MYSQL *db, char *fmt, ...)
{
  va_list ap;
  va_start(ap, fmt);
  vfprintf(stderr, fmt, ap);
  va_end(ap);
  (void)putc('\n', stderr);
  if (db)
    db_disconnect(db);
  exit(EXIT_FAILURE);
}

MYSQL *
db_connect(const char *dbname)
{
  MYSQL *db = mysql_init(NULL);
  if (!db)
    die(db, "mysql_init a èchouè : pas de mèmoire");
  /*
   * Notez que le client et le serveur utilisent des noms de groupes sèparès.
   * Ceci est critique, car le serveur n'acceptera pas les options du client
```

```
   * et vice versa.
   */
  mysql_options(db, MYSQL_READ_DEFAULT_GROUP, "test_libmysqld_CLIENT");
  if (!mysql_real_connect(db, NULL, NULL, NULL, dbname, 0, NULL, 0))
    die(db, "mysql_real_connect a èchouè : %s", mysql_error(db));

  return db;
}

void
db_disconnect(MYSQL *db)
{
  mysql_close(db);
}

void
db_do_query(MYSQL *db, const char *query)
{
  if (mysql_query(db, query) != 0)
    goto err;

  if (mysql_field_count(db) > 0)
  {
    MYSQL_RES   *res;
    MYSQL_ROW    row, end_row;
    int num_fields;

    if (!(res = mysql_store_result(db)))
      goto err;
    num_fields = mysql_num_fields(res);
    while ((row = mysql_fetch_row(res)))
    {
      (void)fputs(">> ", stdout);
      for (end_row = row + num_fields; row < end_row; ++row)
        (void)printf("%s\t", row ? (char*)*row : "NULL");
      (void)fputc('\n', stdout);
    }
    (void)fputc('\n', stdout);
  }
  else
    (void)printf("Lignes affectèes : %lld\n", mysql_affected_rows(db));

  return;

err:
  die(db, "db_do_query a èchouè : %s [%s]", mysql_error(db), query);
}
```

‘GNUmakefile’

```
# On suppose que MySQL est installè dans /usr/local/mysql
inc      := /usr/local/mysql/include/mysql
lib      := /usr/local/mysql/lib

# Si vous n'avez pas encore installè MySQL, essayez plutôt ceci
#inc      := $(HOME)/mysql-4.0/include
#lib      := $(HOME)/mysql-4.0/libmysqld

CC       := gcc
CPPFLAGS := -I$(inc) -D_THREAD_SAFE -D_REENTRANT
CFLAGS   := -g -W -Wall
LDFLAGS  := -static
# Vous pouvez changer -lmysqld en -lmysqlclient pour utiliser
# la librairie client/serveur
LDLIBS    = -L$(lib) -lmysqld -lz -lm -lcrypt

ifneq (,$(shell grep FreeBSD /COPYRIGHT 2>/dev/null))
# FreeBSD
LDFLAGS += -pthread
else
# Linux
LDLIBS += -lpthread
endif

# Cela fonctionne pour les programes de tests sur un simple fichier
sources := $(wildcard *.c)
objects := $(patsubst %c,%o,$(sources))
targets := $(basename $(sources))

all: $(targets)

clean:
rm -f $(targets) $(objects) *.core
```

8.4.9.7 Licence du serveur embarquè

Le code source de MySQL est couvert par la licence GNU GPL (voir Annexe H [GPL license], page 832). L'un des effets de cela est que tout programme qui inclut, par liaison avec `libmysqld`, le code source de MySQL doit être rèalisè en tant que logiciel libre (sous une licence compatible avec la GPL).

Nous encourageons tout le monde á promouvoir le logiciel libre en rèalisant du code sous la licence GPL ou une autre licence compatible. Pour ceux qui ne peuvent le faire, une autre option est d'acheter la licence commerciale pour le code MySQL chez MySQL AB. Pour plus de dètails, voyez Section 1.4.3 [MySQL licenses], page 17.

8.5 Interfaces MySQL pour C++

Deux interfaces sont disponibles dans le dossier des contribution MySQL (`http://www.mysql.com/Downloads/Contrib/`).

8.5.1 Borland C++

Vous pouvez compiler la source Windows de MySQL avec Borland C++ 5.02. (La source Windows n'inclut que les fichiers issus de Microsoft VC++, pour Borland C++ vous devrez crèer les fichiers de projet par vous-même.)

Un probléme connu avec Borland C++ est qu'il utilise un alignement de structures diffèrent de VC++. Cela signifie que vous aurez des problémes si vous essayez d'utiliser la librairie par dèfaut `libmysql.dll` (qui a ètè compilèe avec VC++) avec Borland C++. Vous pouvez faire ce qui suit pour èviter ce probléme.

- Vous pouvez utiliser les librairies MySQL statiques pour Borland C++ que vous trouverez sur `http://www.mysql.com/downloads/os-win32.html`.
- Appelez `mysql_init()` avec `NULL` comme arguement, et non une structure MYSQL prè-allouèe.

8.6 Connectivitè Java/MySQL (JDBC)

Il y'a 2 pilotes JDBC supportès pour MySQL (le pilote Connector/J et le pilote Resin JDBC). Vous pouvez trouver une copie du pilote Connector/J sur `http://www.mysql.com/products/connector-j/` et le pilote Resin sur `http://www.caucho.com/projects/jdbc-mysql/index.xtp`

Pour de la documentation, consultez celle de JDBC et des pilotes pour les fonctionnalitès relatives á MySQL.

8.7 Interface Python pour MySQL

Le dossier des contributions (`http://www.mysql.com/Downloads/Contrib/`) contient une interface Python ècrite par Joseph Skinner.

8.8 Interface Tcl pour MySQL

`http://www.binevolve.com/~tdarugar/tcl-sql/` (Tcl chez binevolve). Le dossier des contributions (`http://www.mysql.com/Downloads/Contrib/`) contient une interface Tcl basèe sur msqltcl 1.50.

8.9 Couche MySQL pour Eiffel

Le dossier des contributions MySQL (`http://www.mysql.com/Downloads/Contrib/`) contient une couche pour Eiffel ècrite par Michael Ravits.

9 Etendre MySQL

9.1 Rouages de MySQL

Ce chapitre dècrit un grand nombre de notions que vous devez connaître lorsque vous travaillez sur le code de MySQL. Si vous envisagez de contribuer au dèveloppement de MySQL, que vous voulez accèder á du code ultra rècent, ou que vous souhaitez simplement vous tenir au courant du dèveloppement, suivez les instructions de la section Section 2.3.4 [Installing source tree], page 91. Si vous êtes intèressès par les rouages internes de MySQL, il est recommandè de vous inscrire á notre liste de diffusion interne `internals`. Cette liste est relativement calme. Pour les dètails d'inscription, voyez Section 1.6.2.1 [Mailing-list], page 24. Tous les dèveloppeurs de MySQL AB sont sur la liste `internals` et nous aidons ceux qui travaillent sur le code de MySQL. Utilisez cette liste pour poser des questions sur le code, et partagez vos patchs que vous voulez voir intègrè au projet MySQL.

9.1.1 Threads MySQL

Le serveur MySQL crèe les threads suivants :

- Le thread de connexion TCP/IP, qui gére toutes les demandes de connexion, et crèe un nouveau thread dèdiè pour gèrer l'identification et le traitement des requêtes SQL, pour chaque connexion.
- Sur Windows NT, il y a un thread appelè gestionnaire de pipe, qui effectue le même travail que le thread de gestion des demandes de connexion, sur les demandes de connexion par pipe.
- Le threads de signal gére tous les signaux. Ce thread gére aussi normalement les alertes et les appels á `process_alarm()` pour forcer les dèlais d'expiration des connexions qui sont inactives.
- Si `mysqld` est compilè avec l'option `-DUSE_ALARM_THREAD`, un thread dèdiè aux alarmes est crèè. Il est uniquement utilisè sur certains systémes oú il y a des problémes avec la fonction `sigwait()` ou si vous voulez utiliser la fonction `thr_alarm()` dans des applications qui n'ont pas de thread dèdiè aux signaux.
- Lors de l'utilisation de l'option `--flush_time=#`, un thread dèdiè est crèè pour vider les tables de la mèmoire á intervalle règulier.
- Chaque connexion a son propre thread dèdiè.
- Chaque table diffèrente sur laquelle des `INSERT DELAYED` sont pratiquèes reçoit un thread.
- Si vous utilisez l'option `--master-host`, un thread de rèplication sera dèmarrè pour lire et appliquer les modifications du maître.

`mysqladmin processlist` affiche uniquement les threads de connexion, ceux de `INSERT DELAYED` et ceux de rèplication.

9.1.2 Suite de test de MySQL

Jusqu'á rècemment, notre suite de test principale ètaient basèe sur des donnèes propriètaires de client, et pour cette raison, il n'a jamais ètè publiè. Le seul systéme de test public actuel est notre script `crash-me`, qui est un script Perl DBI/DBD qui se trouve dans le dossier `sql-bench`, et divers tests qui font parti du dossier `tests`. Le manque d'une suite de test publique et standardisèe rend difficile á nos utilisateurs et nos dèveloppeurs les possibilitès de tests de règressions. Pour corriger ce probléme, nous avons crèè un nouveau systéme de tests qui est inclus dans les distributions source et binaires, á partir de la version 3.23.29.

Le jeu de test actuel ne couvre pas toutes les situations en MySQL, mais il permet d'identifier les bogues les plus courants lors de requêtes SQL, les problémes de librairies et aussi les problémes de rèplication. Notre but final est de lui faire couvrir 100% du code. Nous apprècions les contributions á notre suite de test. Vous pouvez notamment fournir des test qui examine certaines fonctions critiques de votre systéme, et qui assureront que les futures versions de MySQL le prennent en compte.

9.1.2.1 Exècuter la suite de tests MySQL

Le systéme de tests est constitutè d'un interprèteur de langage de tests (`mysqltest`), un script shell qui exècute tous les scripts (`mysql-test-run`), les cas de tests rèels, ècrits dans un langage spècial de tests, et leur rèsultats attendus. Pour exècuter ces tests sur votre systéme aprés une compilation, tapez `make test` ou `mysql-test/mysql-test-run` depuis la racine de la distribution. Si vous avez installè une distribution binaire, `cd` jusqu'au dossier d'installation (eg. `/usr/local/mysql`), et exècutez `scripts/mysql-test-run`. Tous les tests doivent rèussir. Si ce n'est pas le cas, vous devriez essayer de trouver pourquoi, et faire un rapport de bogues á MySQL. Voir Section 9.1.2.3 [Reporting mysqltest bugs], page 671.

Si vous avez une copie de `mysqld` qui fonctionne sur la machine oú vous voulez faire des tests, vous n'avez pas á l'arrêter, tant qu'elle n'utilise pas les ports `9306` et `9307`. Si l'un de ces ports est pris, vous devriez èditer le script `mysql-test-run` et changer les valeurs des ports des maîtres et esclaves, en les remplaçant par des ports libres.

Vous pouvez exècuter des tests individuels avec `mysql-test/mysql-test-run test_name`.

Si l'un des tests èchoue, vous devriez tester `mysql-test-run` avec l'option `--force` pour vèrifier si aucun autre test n'èchoue.

9.1.2.2 Amèliorer la suite de tests MySQL

Vous pouvez utiliser le langage de `mysqltest` pour ècrire vos propres cas de tests. Malheureusement, nous n'avons pas encore ècrit une documentation compléte pour ce logiciel, et nous prèvoyons de le faire rapidement. Vous pouvez, toutefois, utiliser les cas de tests actuels comme exemples. Les points suivants devraient vous mettre le pied á l'ètrier.

- Les tests sont situè dans `mysql-test/t/*.test`
- Un cas de tests est constituè de commandes terminèes par un `;`, et est similaire aux donnèes d'entrèes du client `mysql`. Une commande est par dèfaut une commande

envoyèe au serveur MySQL, á moins qu'il ne soit reconnu comme une commande interne (par exemple, `sleep`).

- Toutes les requêtes qui produisent des rèsultats, comme `SELECT`, `SHOW`, `EXPLAIN`, etc., doivent être prècèdèes par `@/path/to/result/file`. Le fichier contient alors les rèsultats attendus. Un moyen simple pour gènèrer le rèsultat du fichier est d'exècuter `mysqltest -r < t/test-case-name.test` depuis le dossier de tests `mysql-test`, puis d'èditer le fichier rèsultant, si nècessaire, pour ajuster le contenu. Dans ce cas, soyez trés prudent lors de l'ajout ou la suppression de caractéres invisibles : assurez vous de ne changer que du texte, ou d'effacer des lignes. Vous pouvez utiliser `od -c` pour vous assurer que votre èditeur n'a pas perturbè le fichier durant l'èdition. Bien sur, nous espèrons que vous n'aurez jamais a èditer le rèsultat du fichier `mysqltest -r` ca vous n'avez á faire cela que lorsque vous dècouvrez un bug.
- Pour être cohèrent avec votre configuration, vous devriez placer les fichiers de rèsultats dans le dossier `mysql-test/r` et les nommer `test_name.result`. Si le test produit plus qu'un rèsultat, vous devez utiliser `test_name.a.result`, `test_name.b.result`, etc.
- Si une commande retourne une erreur, vous devez, sur la ligne de la commande, le spècifier avec `--error error-number`. Le numèro d'erreur peut être une liste d'erreurs possibles, sèparèes par des virgules `','`.
- Si vous ècrivez un test de rèplication, vous devez, sur la premiére ligne du fichier de test, ajouter le code `source include/master-slave.inc;`. Pour passer entre le maître et l'esclave, utilisez `connection master;` et `connection slave;`. Si vous avez besoin d'utiliser une connexion alternative, vous pouvez utiliser `connection master1;` pour le maître, et `connection slave1;` pour l'esclave.
- Si vous avez besoin d'une boucle, vous pouvez utiliser ceci :

```
let $1=1000;
while ($1)
{
 # votre requête ici
 dec $1;
}
```

- Pour faire une pause entre les requêtes, utilisez la commande `sleep`. Elle supporte les fraction de secondes, ce qui vous permet d'utiliser `sleep 1.3;`, pour attendre 1,3 secondes.
- Pour exècuter l'esclave avec des options additionnelles pour votre cas de tests, ajoutez les au format ligne de commande dans `mysql-test/t/test_name-slave.opt`. Pour le maître, ajoutez les dans `mysql-test/t/test_name-master.opt`.
- Si vous avez une question sur la suite de tests, ou que vous avez un test á proposer, envoyez le par email á `internals@lists.mysql.com`. Comme la liste n'accepte pas les attachements, vous devriez les placer sur le serveur FTP : `ftp://support.mysql.com/pub/mysql/Incoming/`

9.1.2.3 Rapporter des bugs dans la suite de tests MySQL

Si votre versionn de MySQL ne passe pas un teste, vous devez faire ceci :

- N'envoyez pas de rapport de bug avant d'avoir ètudier au maximum les raisons possibles de l'èchec! Lorsque vous le faîtes, utilisez le programme `mysqlbug`, pour que nous puissions obtenir un maximum d'informations sur votre systéme et la version de `MySQL`. Voir Section 1.6.2.3 [Bug reports], page 27.
- Assurez vous d'inclure le rèsultat de `mysql-test-run`, ainsi que le contenu de tous les fichiers `.reject` du dossier `mysql-test/r`.
- Si un test de la suite èchoue, vèrifiez si le test èchoue aussi en l'exècutant seul :

  ```
  cd mysql-test
  mysql-test-run --local test-name
  ```

  Si cela èchoue, alors vous devriez configurer MySQL avec `--with-debug` et exècuter `mysql-test-run` avec l'option `--debug`. Si cela èchoue aussi, envoyez le fichier de trace '`var/tmp/master.trace`' á `ftp://support.mysql.com/pub/mysql/secret` pour que nous puissions l'examiner. N'oubliez pas d'inclure une description compléte de votre systéme, ainsi que de la version de l'exècutable `mysqld`, et de sa compilation.
- Essayez d'exècuter `mysql-test-run` avec l'option `--force` pour voir si il n'y a pas d'autres tests qui èchouent.
- Si vous avez compilè MySQL vous-même, vèrifiez notre manuel, ainsi que les notes de compilations pour votre plate-forme, ou bien, utilisez á la place un des exècutables que nous avons compilè pour vous, disponibles á `http://www.mysql.com/downloads/`. Toutes nos versions exècutables doivent passer la suite de tests.
- Si vous obtenez une erreur, comme `Result length mismatch` ou `Result content mismatch`, cela signifie que le rèsultat de la suite de tests n'a pas la taille attendue. Cela peut être un bug de MySQL, ou que votre version de MySQL fournit un rèsultat d'une autre taille, dans certaines circonstances.

  Les rèsultats de tests qui ont èchouès sont placès dans un fichier avec le même nom de base que le fichier de test, et avec l'extension `.reject`. Si votre test èchoue, faites un `diff` sur les deux fichiers. Si vous ne pouvez pas voir oú ils différent, examinez ces deux fichiers avec `od -c`, et vèrifiez leur tailles respectives.
- Si un test èchoue totalement, vous devriez vèrifier les fichiers de log dans le dossier `mysql-test/var/log`, pour avoir des indices sur ce qui a èchouè.
- Si vous avez compilè MySQL avec le dèbogage, vous pouvez essayer de le dèboger en exècutant `mysql-test-run` avec `--gdb` et/ou `--debug`. Voir Section E.1.2 [Making trace files], page 816.

  Si vous n'avez pas compilè MySQL pour le dèbogage, vous devriez essayer de le faire. Spècifiez simplement l'option `--with-debug` dans le script de `configure`! Voir Section 2.3 [Installing source], page 84.

9.2 Ajouter des fonctions á MySQL

Il y a deux mèthodes pour ajouter des fonctions á MySQL :

- Vous pouvez ajouter la fonction grâce á l'interface de fonctions utilisateur (UDF). Les fonctions utilisateur sont ajoutèes et supprimèes dynamiquement avec les commandes `CREATE FUNCTION` et `DROP FUNCTION`. Voir Section 9.2.1 [`CREATE FUNCTION`], page 673.

- Vous pouvez ajouter une fonction sous la forme native (intègrèe) d'une fonction MySQL. Les fonctions natives sont compilèes dans `mysqld` et sont disponibles en permanence.

Chaque mèthode a ses avantages et inconvènients :

- Si vous ècrivez une fonction utilisateur, vous devez installer le fichier objet en plus du serveur lui-même. Si vous compilez votre fonction dans le serveur, vous n'avez pas ce probléme.
- Vous pouvez ajouter des UDF á une distribution binaire de MySQL. Les fonctions natives requiérent une modification de la distribution source.
- Si vous mettez á jour votre distribution MySQL, vous pouvez continuer á utiliser vos fonctions prècèdemment installèes. Pour les fonctions natives, vous devez refaire les modifications du code á chaque mise á jour.

Quelque soit la mèthode que vous utilisez pour ajouter de nouvelles fonctions, ces fonctions pourront être utilisèes comme des fonctions natives telles que `ABS()` ou `SOUNDEX()`.

9.2.1 Syntaxe de `CREATE FUNCTION/DROP FUNCTION`

```
CREATE [AGGREGATE] FUNCTION nom_fonction RETURNS {STRING|REAL|INTEGER}
       SONAME nom_librairie_partagèe

DROP FUNCTION nom_fonction
```

Une fonction dèfinie par un utilisateur (UDF) est une mèthode pour intègrer une fonction qui fonctionne de la même façon qu'une fonction native de MySQL, comme `ABS()` et `CONCAT()`.

`AGGREGATE` est une nouvelle option pour MySQL version 3.23. Une fonction `AGGREGATE` fonctionne exactement comme une fonction native comme `SUM` ou `COUNT()`.

`CREATE FUNCTION` enregistre le nom de la fonction, le type, et le nom des librairies partagèes dans la table `mysql.func`. Vous devez avoir les droits `INSERT` et `DELETE` dans la base `mysql` pour crèer et supprimer les fonctions.

Toutes les fonctions actives sont rechargèes chaque fois que le serveur dèmarre, sauf si vous dèmarrez `mysqld` avec l'option `--skip-grant-tables`. Dans ce cas, l'utilisation des UDF n'est pas prise en compte et les UDFs ne sont pas disponibles. (Une fonction active est une fonction qui peut être chargèe avec `CREATE FUNCTION` et supprimèe par `REMOVE FUNCTION`).

Concernant l'ècriture des UDFs, Section 9.2 [Adding functions], page 672. Pour que le mècanisme des fonctions UDF fonctionne, les fonctions doivent être ècrites en C ou C++, votre systéme doit supporter le chargement dynamique et vous devez avoir compilè `mysqld` dynamiquement (pas statiquement).

Notez que pour faire fonctionner `AGGREGATE`, vous devez avoir une table `mysql.func` qui contient la colonne `type`. Si ce n'est pas le cas, vous devez exècuter le script `mysql_fix_privilege_table` pour rèsoudre ce probléme.

9.2.2 Ajouter une nouvelle fonction dèfinie par l'utilisateur (UDF)

Pour que le mècanisme UDF fonctionne, les fonctions doivent êtres ècrites en C ou C++ et votre systéme doit supporter le chargement dynamique. Les sources de MySQL incluent un

fichier '`sql/udf_example.cc`' qui dèfinit 5 nouvelles fonctions. Consultez ce fichier pour voir comment marchent les conventions d'appels des UDF.

Pour que `mysqld` puisse utiliser les fonctions UDF, vous devez configurer MySQL avec l'option `--with-mysqld-ldflags=-rdynamic`. La raison est que sur diverses plate-formes, (Linux inclus) vous pouvez charger une librairie dynamique (avec `dlopen()`) depuis un programme statique liè, que vous pouvez obtenir si vous utilisez l'option `--with-mysql-ldflags=-all-static`. Si vous voulez utiliser une UDF qui nècessite un accés aux symboles de `mysqld` (comme l'exemple `methaphone` dans '`sql/udf_example.cc`' qui utilise `default_charset_info`), vous devez lier le programme avec `-rdynamic` (voir `man dlopen`).

Pour chaque fonction que vous voulez utiliser dans SQL, vous devez dèfinir les fonctions correspondantes en C (ou C++). Dans la discussion ci-dessous, le nom "xxx" est utilisè comme un exemple de nom de fonction. Pour faire la diffèrence entre l'usage de SQL et de C/C++, `XXX()` (majuscules) indique l'appel d'une fonction SQL et `xxx()` (minuscules) indique l'appel d'une fonction C/C++.

Les fonctions C/C++ que vousècrivez pour l'implèmentation de l'interface de `XXX()` sont :

`xxx()` (requis)

La fonction principale. C'est lá oú le rèsultat de la fonction est calculè. La correspondance entre le type de SQL et le type retournè par votre fonction C/C++ est affichè ci-dessous :

SQL type	**C/C++ type**
`STRING`	`char *`
`INTEGER`	`long long`
`REAL`	`double`

`xxx_init()` (optionnel)

La fonction d'initialisation de `xxx()`. Elle peut-être utilisèe pour :

- Vèrifier le nombre d'arguments de `XXX()`.
- Vèrifier que les arguments correspondent aux types requis ou indiquer á MySQL de contraindre des arguments aux types que vous voulez quand la fonction principale est appelèe.
- Allouer la mèmoire requise pour la fonction principale.
- Spècifier la longueur maximale de la sortie.
- Spècifier (pour les fonctions `REAL`) le nombre maximal de dècimales.
- Spècifier si le rèsultat peut-être `NULL`.

`xxx_deinit()` (optionnel)

La dès-initialisation de la fonction `xxx()`. Elle doit dès-allouer toute la mèmoire allouèe par l'initialisation de la fonction.

Quand une requête SQL fait appel á `XXX()`, MySQL appelle l'initialisation de la fonction `xxx_init()`, pour laisser exècuter n'importe quelle action exigèe, telle que la vèrification d'arguments ou l'allocation de mèmoire.

Si `xxx_init()` retourne une erreur, la requête SQL est annulèe avec un message d'erreur et la fonction principale et la fonction de dès-initialisation ne sont pas appelèes. Autrement, la fonction principale `xxx()` est appelèe une fois pour chaque ligne. Aprés que toutes les lignes

aient ètè traitèes, la fonction de dès-initialisation `xxx_deinit()` est appelèe pour procèder aux nettoyages requis.

Pour les fonctions d'agrègat (comme `SUM()`), vous pouvez ègalement ajouter les fonctions suivantes :

`xxx_reset()` (requise)
: Remet la somme á zèro et insére l'argument en tant que valeur initiale pour un nouveau groupe.

`xxx_add()` (requise)
: Ajoute l'argument á l'ancienne somme.

Quand vous utilisez les UDF d'agrègat, MySQL opére comme suit :

All functions must be thread-safe (not just the main function, but the initialisation and deinitialisation functions as well). This means that you are not allowed to allocate any global or static variables that change! If you need memory, you should allocate it in `xxx_init()` and free it in `xxx_deinit()`.

1. Appeler `xxx_init()` pour laisser la fonction d'agrègat allouer la mèmoire dont elle aura besoin pour stocker les rèsultats.
2. Trier la table en accord avec la clause `GROUP BY`.
3. Pour la premiére ligne dans un nouveau groupe, appeler la fonction `xxx_reset()`.
4. Pour chaque ligne appartenant á un même groupe, appeler la fonction `xxx_add()`.
5. Quand le groupe change ou lorsque la derniére ligne a ètè traitèe, appeler `xxx()` pour obtenir le rèsultat de l'agrègat.
6. Rèpèter 3-5 tant que toutes les lignes n'ont pas ètè traitèes.
7. Appeler `xxx_deinit()` pour libèrer la mèmoire allouèe.

Toutes les fonctions doivent être sûrs pour les threads (pas seulement la fonction principale, mais les fonctions d'initialisation et de dès-initialisation ègalement). Cela signifie que vous n'êtes pas autorisès á allouer une variable globale ou statique qui change ! Si vous avez besoin de mèmoire, vous devez l'allouer avec la fonction `xxx_init()` et la libèrer avec `xxx_deinit()`.

9.2.2.1 Fonctions utiliateur : appeler des fonctions simples

La fonction principale doit être dèclarèe comme illustrèe ici. Notez que le type de retour et les paramétres différent, suivant que vous voulez dèclarer une fonction SQL `XXX()` qui retournera une `STRING`, un `INTEGER` ou un `REAL` dans la commande `CREATE FUNCTION` :

Pour les fonctions de chînes `STRING` :

```
char *xxx(UDF_INIT *initid, UDF_ARGS *args,
          char *result, unsigned long *length,
          char *is_null, char *error);
```

Pour les fonctions d'entiers `INTEGER` :

```
long long xxx(UDF_INIT *initid, UDF_ARGS *args,
              char *is_null, char *error);
```

Pour les fonctions de nombres á virgule flottante `REAL` :

```
double xxx(UDF_INIT *initid, UDF_ARGS *args,
           char *is_null, char *error);
```

Les fonctions d'initialisation et de terminaison sont dèclarèes comme ceci :

```
my_bool xxx_init(UDF_INIT *initid, UDF_ARGS *args, char *message);

void xxx_deinit(UDF_INIT *initid);
```

Le paramétre `initid` est passè aux trois fonctions. Il pointe sur une structure `UDF_INIT` qui est utilisèe pour communiquer des informations entre les fonctions. Les membres de la structure `UDF_INIT` sont ceux qui sont listès ci-dessous. La fonction d'initialisation doit prèparer les membres qu'elle veut, et notamment leur donner une valeur initiale (pour utiliser les valeurs par dèfaut, laissez le membre intact) :

`my_bool maybe_null`
: `xxx_init()` doit remplacer `maybe_null` par `1` si `xxx()` peut retourner `NULL`. La valeur par dèfaut est `1` si l'un des arguments n'est dèclarè comme `maybe_null`.

`unsigned int decimals`
: Le nombre de dècimales. La valeur par dèfaut est le nombre maximum de dècimales dans l'argument passè á la fonction. Par exemple, vis la fonction reçoit `1.34`, `1.345`, and `1.3`, ce nombre sera 3, car `1.345` a 3 dècimales.

`unsigned int max_length`
: La taille maximale de la chaîne rèsultat. La valeur par dèfaut dèpend du type de rèsultat de la fonction. Pour les fonctions de chaînes, la valeur par dèfaut est la taille du plus grand argument. Pour les fonctions entiéres, la valeur est de 21 chiffres. Pour les fonctions á nombre á virgule flottante, la valeur est de 13, plus le nombre de dècimales indiquèes par `initid->decimals`. Pour les fonctions numèriques, la taille inclut le signe et le sèparateur dècimal.

  Si vous voulez retourner un BLOB, vous devez donner á ce membre la valeur de 65K ou 16M; cet espace mèmoire ne sera pas allouè, mais utilisè pour dècider quel type de colonne utiliser, si il y a un besoin de stockage temporaire.

`char *ptr` Un pointeur que la fonction peut utiliser pour ses besoins propres. Par exemple, la fonction peut utiliser `initid->ptr` pour transfèrer de la mèmoire allouèe entre les trois fonctions. En `xxx_init()`, allouez de la mèmoire, et assignez la á ce pointeur :

  ```
  initid->ptr = allocated_memory;
  ```

  En `xxx()` et `xxx_deinit()`, utilisez `initid->ptr` pour exploiter ou supprimer la mèmoire.

9.2.2.2 Appeler des fonctions utilisateurs pour les groupements

Voici une description des diffèrentes fonctions que vous devez dèfinir pour rèaliser des calculs sur des regroupements, avec une fonction utilisateur :

```
char *xxx_reset(UDF_INIT *initid, UDF_ARGS *args,
                char *is_null, char *error);
```

Cette fonction est appelèe lorsque MySQL trouve la premiére ligne dans un nouveau groupe. Dans cette fonction, vous devez remettre á zèro des variables internes de sommaire, puis indique le nouvel argument comme premier membre du nouveau groupe.

Dans de nombreuses situations, cela se fait en interne en remettant á zèro toutes les variables, et en appelant `xxx_add()`.

```
char *xxx_add(UDF_INIT *initid, UDF_ARGS *args,
              char *is_null, char *error);
```

Cette fonction est appelèe á chaque fois qu'une ligne qui appartient á un groupe est trouvèe, hormis la premiére ligne. Durant cette fonction, vous devez ajouter les donnèes dans votre variable interne de sommaire.

La fonction `xxx()` doit être dèclarèe de maniére identique á celle d'un fonction utilisateur simple. Voir Section 9.2.2.1 [UDF calling], page 675.

Cette fonction est appelèe lorsque toutes les lignes d'un groupe ont ètè traitèes. Vous ne devez normalement pas accèder á la variable `args` ici, mais retourner votre valeur, á partir des valeurs du sommaire interne.

Tous les traitements des arguments de `xxx_reset()` et `xxx_add()` doivent être fait d'une maniére similaire á celle des fonctions UDF normales. Voir Section 9.2.2.3 [UDF arguments], page 677.

La gestion de la valeur retournèe par `xxx()` doit être identique á celle d'une fonction utilisateur classique. Voir Section 9.2.2.4 [UDF return values], page 679.

Le pointeur argument de `is_null` et `error` sont les mêmes pour tous les appels de `xxx_reset()`, `xxx_add()` et `xxx()`. Vous pouvez utiliser ces valeur pour vous rappeler si vous avez rencontrè une erreur, ou si la fonction `xxx()` doit retourner `NULL`. Notez que vous ne devez pas stocker de chaîne dans `*error`! C'est un conteneur d'un seul octet!

`is_null` est remis á zèro pour chaque (avant d'appeler `xxx_reset()`. `error` n'est jamais remis á zèro.

Si `isnull` ou `error` sont modifiès aprés `xxx()`, alors MySQL va retourner `NULL` comme rèsultat de la fonction de groupement.

9.2.2.3 Traitement des arguments

Le paramétre `args` pointe sur une structure `UDF_ARGS` qui dispose des membre suivants :

`unsigned int arg_count`
: Le nombre d'arguments. Vèrifiez cette avleur dans la fonction d'initialisation, si vous voulez que votre fonction soit appelèe avec un nombre particulier d'arguments. Par exemple :

```
if (args->arg_count != 2)
{
    strcpy(message,"XXX() requires two arguments");
    return 1;
}
```

`enum Item_result *arg_type`
: Le type de chaque argument. Les valeurs possibles pour chaque type sont `STRING_RESULT`, `INT_RESULT` et `REAL_RESULT`.

Pour s'assurer que les arguments sont d'un type donnè, et retourner une erreur dans le cas contraire, vèrifiez le tableau `arg_type` durant la fonction d'initialisation. Par exemple :

```
if (args->arg_type[0] != STRING_RESULT ||
    args->arg_type[1] != INT_RESULT)
{
    strcpy(message,"XXX() requires a string and an integer");
    return 1;
}
```

Comme alternative á l'imposition d'un type particulier pour les arguments des focntiosn, vous pouvez utiliser la fonction pour qu'elle modifie le type des arguments et donne aux valeurs de `arg_type` le type que vous souhaitez. Cela fait que MYSQL va forcer les arguments á un type donnèe, pour chaque appel de la fonction `xxx()`. Par exemple, pour forcer le type des deux premiers arguments en chaîne et entier, vous pouvez utiliser la fonction d'initisalition `xxx_init()` :

```
args->arg_type[0] = STRING_RESULT;
args->arg_type[1] = INT_RESULT;
```

`char **args`

`args->args` communique les informations á la fonction d'initialisation, ainsi que la nature des arguments avec laquelle elle a ètè appelèe. Pour un argument constant `i`, `args->args[i]` pointe sur la valeur de l'argument. Voir plus bas pour les instructions d'accés á cette valeur. Pour les valeurs non constantes, `args->args[i]` vaut `0`. Un argument constant est une expression qui utilise des constantes, comme `3` ou `4*7-2` ou `SIN(3.14)`. Un argument non-constant est une expression qui fait rèfèrence aux valeurs qui peuvent changer de ligne en ligne, par exemple des noms de colonnes ou des fonctions qui sont appelèes avec des arguments non-constants.

Pour chaque invocation de la fonction principale, `args->args` contient les arguments rèels qui sont passès á la ligne qui sera traitèe.

Les fonctions peuvent faire rèfèrence á un argument `i` comme ceci :

- Un argument de type `STRING_RESULT` est donnè sous la forme d'un pointeur de chaîne, plus une longueur, pour permettre la gestion des donnèes binaires ou des donnèes de taille arbitraire. Le contenu des chaînes est disponible avec l'expression `args->args[i]` et la taille de la chaîne est `args->lengths[i]`. Ne supposez pas que les chaînes sont terminès par le caractére nul.
- Pour un argument de type `INT_RESULT`, vous devez transtyper la valeur `args->args[i]` en valeur `long long` :

  ```
  long long int_val;
  int_val = *((long long*) args->args[i]);
  ```

- Pour un argument de type `REAL_RESULT`, vous devez transtyper la valeur `args->args[i]` en valeur `double` :

  ```
  double    real_val;
  real_val = *((double*) args->args[i]);
  ```

`unsigned long *lengths`

Pour une fonction d'initialisation, le tableau `lengths` indique la taille maximale des chaînes pour chaque argument. Vous ne devez pas les modifier. Pour chaque appel de la fonction principale, `lengths` contient la taille rèelle de toutes les chaînes arguments qui sont passè pour la ligne traitèe. Pour les arguments de type `INT_RESULT` ou `REAL_RESULT`, `lengths` contient toujours la taille maximale de l'argument (comme pour la fonction d'initialisation).

9.2.2.4 Valeurs de retour et gestion d'erreurs.

La fonction d'initialisation doit retourner `0` si aucune erreur ne s'est produite et `1` sinon. Si une erreur s'est produite, `xxx_init()` doit stocker un message se terminant par un NULL dans le paramétre `message`. Le message sera retournè au client. La taille du tampon du message est de `MYSQL_ERRMSG_SIZE` caractéres, mais vous devez essayer de garder une taille de message infèrieure á 80 caractéres, sinon, il remplit la largeur d'un ècran de terminal standard.

La valeur de retour de la fonction principale `xxx()` est la valeur de la fonction, pour les fonctions `long long` et `double`. Une fonction de chaîne de caractéres doit retourner un pointeur vers le rèsultat et stocker la taille de la chaîne de caractéres dans l'argument `length`.

Affectez cette valeur au contenu et á la longueur de la valeur retournèe. Par exemple :

```
memcpy(result, "chaîne retournèe", 16);
*length = 16;
```

Le tampon `result` qui est passè á la fonction calc á une taille de 255 bits. Si votre rèsultat dèpasse ceci, ne vous inquietez pas de l'allocation de mèmoire pour ce rèsultat.

Si votre fonction de chaînes de caractéres a besoin de retourner une chaîne de caractéres plus grande que 255 bits, vous devez allouer de l'espace pour cela avec `malloc()` dans votre fonction `xxx_init()`. Vous pouvez stocker la mèmoire allouèe dans le slot `ptr` de la structure `UDF_INIT` pour être rè-utilisèe par les appels futurs de `xxx()`. Voir Section 9.2.2.1 [UDF calling], page 675.

Pour indiquer une valeur de retour `NULL` dans la fonction principale, mettez `is_null` á `1` :

```
*is_null = 1;
```

Pour indiquer une erreur retournèe dans la fonction principale, mettez le paramétre `error` á 1:

```
*error = 1;
```

Si `xxx()` met `*error` á `1` pour chaque ligne, la valeur de la fonction est `NULL` pour la ligne en question et pour chaque ligne suivante traitèe par le processus dans lequel `XXX()` est invoquè. (`xxx()` ne sera même pas appelè pour les lignes suivantes.) **Remarque :** dans les versions antèrieures á 3.22.10, vous devez dèfinir `*error` et `*is_null` :

```
*error = 1;
*is_null = 1;
```

9.2.2.5 Compiler et installer des fonctions utilisateurs

Les fichiers qui implèmentent des fonctions utilisateurs doivent être compilès et installès sur le même hôte que celui du serveur. Ce processus est dècrit plus bas, avec le fichier '`udf_example.cc`' qui est inclut dans les sources MysQL. Ce fichier contient les fonctions suivantes :

- `metaphon()` retourne la version mètaphone de la chaîne en argument. C'est une technique proche du soundex, mais elle est bien plus optimisèe pour l'anglais.
- `myfunc_double()` retourne la moyenne des codes ASCII des caractéres de la chaîne passèe en argument.
- `myfunc_int()` retourne la somme de tailles des arguments.
- `sequence([const int])` retourne une sèquence, commençant á partir du nombr choisit ou 1, si aucun nombre n'a ètè fourni.
- `lookup()` retourne l'adresse IP numèrique d'un hôte.
- `reverse_lookup()` retourne le nom d'hôte pour une adresse IP. Cette fonction peut être appelèe avec une chaîne au format `"xxx.xxx.xxx.xxx"` ou quatre nombres.

Un fichier dynamiquement chargeable doit être compilè sous la forme d'un objet partagè, grâce á une commande comme celle-ci :

```
shell> gcc -shared -o udf_example.so myfunc.cc
```

Vous pouvez facilement trouver les options correctes pour la compilation en exècutant cette commande dans le dossier '`sql`' de votre installation source :

```
shell> make udf_example.o
```

Vous devez exècuter une commande de compilation similaire á celle que le `make` affiche, sauf que vous devrez supprimer l'option `-c` prés de la fin de la ligne, et ajouter `-o udf_example.so` á la fin de la ligne. Sur certains systémes, vous devrez aussi supprimer `-c` de la commande).

Une fois que vous compilez un objet partagès contenant des fonctions utilisateurs, vous devez les installer, et prèvenir le serveur MYSQL. Compilier un objet partagè avec '`udf_example.cc`' produit un fichier qui s'appelle '`udf_example.so`' (le nom exact peut varier suivant la plate-forme). Copiez ce fichier dans l'un des dossiers utilisè par `ld`, tel que '`/usr/lib`'. Sur de nombreux systémes, vous pouvez faire pointer la variable d'environnement `LD_LIBRARY` ou `LD_LIBRARY_PATH` pour qu'elle pointe dans le dossier oú vous avez vos fichiers de fonctions. Le manuel de `dlopen` vous indiquera quelle variable utiliser sur votre systéme. Vous devriez indiquer cette valeur dans les options de dèmarrage de `mysql.server` et `safe_mysqld`, et redèmarrer `mysqld`.

Aprés que la librairie ait ètè installèe, indiquez á `mysqld` ces nouvelles fonctions avec ces commandes :

```
mysql> CREATE FUNCTION metaphon RETURNS STRING SONAME "udf_example.so";
mysql> CREATE FUNCTION myfunc_double RETURNS REAL SONAME "udf_example.so";
mysql> CREATE FUNCTION myfunc_int RETURNS INTEGER SONAME "udf_example.so";
mysql> CREATE FUNCTION lookup RETURNS STRING SONAME "udf_example.so";
mysql> CREATE FUNCTION reverse_lookup
    ->        RETURNS STRING SONAME "udf_example.so";
```

```
mysql> CREATE AGGREGATE FUNCTION avgcost
    ->        RETURNS REAL SONAME "udf_example.so";
```

Les fonctions peuvent être effacèes plus tard avec `DROP FUNCTION`:

```
mysql> DROP FUNCTION metaphon;
mysql> DROP FUNCTION myfunc_double;
mysql> DROP FUNCTION myfunc_int;
mysql> DROP FUNCTION lookup;
mysql> DROP FUNCTION reverse_lookup;
mysql> DROP FUNCTION avgcost;
```

Les commandes `CREATE FUNCTION` et `DROP FUNCTION` modifient la table systéme `func` dans la base `mysql`. Le nom de la fonction, son type et le nom de la librairie partagèe sont alors sauvès dans la table. Vous devez avoir les droits de `INSERT` et `DELETE` dans la base `mysql` pour ajouter et effacer des fonctions.

Vous ne devez pas utiliser la commande `CREATE FUNCTION` pour ajouter une fonction qui a dèjá ètè crèèe. Si vous devez reinstaller une fonction, vous devez la supprimer avec la commande `DROP FUNCTION` puis la reinstaller avec `CREATE FUNCTION`. Vous devrez faire cela, par exemple, si vous recompilez une nouvelle version de votre fonction, pour que `mysqld` utilise cette nouvelle version. Sinon, le serveur va continuer á utiliser l'ancienne version.

Les fonctions actives sont rechargèes á chaque fois que le serveur dèmarre, á moins que vous ne dèmarriez le serveur `mysqld` avec l'option `--skip-grant-tables`. Dans ce cas, l'initialisation des fonctions utilisateurs sont ignorèes, et ces fonctions sont inutilisables. Une fonction active doit avoir ètè crèèe avec `CREATE FUNCTION` et pas supprimèe avec `DROP FUNCTION`.

9.2.3 Ajouter de nouvelles fonctions natives

La procèdure pour ajouter une nouvelle fonction native est dècrite ici. Notez que vous ne pouvez ajouter de fonctions natives á une distribution binaire car la procèdure implique la modifications du code source de MySQL. Vous devez compiler MySQL vous-même á partir d'une distribution des sources. Notez aussi que si vous migrez vers une autre version de MySQL (par exemple, quand une nouvelle version est rèalisèe), vous devrez rèpèter la procèdure avec la nouvelle version.

Pour ajouter une nouvelle fonction native á MySQL, suivez les ètapes suivantes :

1. Ajoutez une ligne dans '`lex.h`' qui dèfinit le nom de la fonction dans le tableau `sql_functions[]`.
2. Si le prototype de la fonction est simple (elle prend zèro, un, deux ou trois arguments), vous devez spècifier dans lex.h SYM(FUNC_ARG#) (oú # est le nombre d'arguments) en tant que second argument dans le tableau `sql_functions[]` et ajouter une fonction qui crèe un objet fonction dans '`item_create.cc`'. Regardez `"ABS"` et `create_funcs_abs()` pour un exemple.

   Si le prototype de la fonction est compliquè (par exemple, elle prend un nombre variable d'arguments), vous devez ajouter deux lignes á '`sql_yacc.yy`'. Une qui indique le symbole prèprocesseur que `yacc` doit dèfinir (cela doit être ajoutè au dèbut du fichier). Puis dèfinir les paramétres de la fonction et un "item" avec ces paramétres á

la régle `simple_expr`. Pour un exemple, vèrifiez toutes les occurences de `ATAN` dans '`sql_yacc.yy`' pour voir comment cela se fait.

3. Dans '`item_func.h`', dèclarez une classe hèritant de `Item_num_func` ou de `Item_str_func`, selon que vvotre fonction retourne une nombre ou un chaîne.
4. Dans le fichier '`item_func.cc`', ajoutez l'une des dèclaration suivantes, selon que vous dèfinissez une fonction numèrique ou de chaîne de caractéres :

   ```
   double   Item_func_newname::val()
   longlong Item_func_newname::val_int()
   String  *Item_func_newname::Str(String *str)
   ```

   Si vous hèritez votre objet de l'un des èlèments standards (comme `Item_num_func`) vous n'aurez probablement qu'á dèfinir l'une des fonctions dècrites ci-dessus et laisser l'objet parent prendre soin des autres fonctions. Par exemple, la classe `Item_str_func` dèfinit une fonction `val()` qui exècute `atof()` sur la valeur retournèe par `::str()`.
5. Vous devez aussi probablement dèfinir la fonction objet suivante :

   ```
   void Item_func_newname::fix_length_and_dec()
   ```

   Cette fonction doit au moins calculer `max_length` en se basant sur les arguments donnès. `max_length` est le nombre maximal de caractéres que la fonction peut retourner. Cette fonction doit aussi dèfinir `maybe_null = 0` si la fonction principale ne peut pas retourner une valeur `NULL`. La fonction peut vèrifier si l'un de ses arguments peut retourner `NULL` en vèrifiant la variable `maybe_null` des arguments. Vous pouvez regarder `Item_func_mod::fix_length_and_dec` pour avoir un exemple concret.

Toutes les fonctions doivent être sûres pour les threads (en d'autres termes, n'utilisez aucune variable statique ou globale dans la fonction sans les protèger avec mutexes).

Si vous voulez retourner `NULL`, á partir de `::val()`, `::val_int()` ou `::str()` vous devez mettre `null_value` á 1 et retourner 0.

Pour les fonctions de l'objet `::str()`, il y'a d'autre considèrations á prendre en compte :

- L'argument `String *str` fournit un tampon de chaîne qui peut être utilisè pour contenir le rèsultat. (Pour plus d'informations á propos du type `String`, regardez le fichier '`sql_string.h`'.)
- La fonction `::str()` doit retourner la chaîne contenant le rèsultat ou `(char*) 0` si celui-ci est `NULL`.
- Aucune des fonctions de chaînes n'essaye d'allouer de mèmoire tant que ce n'est pas nècessaire !

9.3 Ajouter une nouvelle procèdure á MySQL

Avec MySQL, vous pouvez dèfinir une procèdure en C++ qui accéde et modifie les donnèes dans la requête avant que celle-ci ne soit envoyèe au client. La modification peut être faite ligne par ligne ou au niveau de `GROUP BY`.

Nous avons crèe une procèdure d'exemple avec la version 3.23 de MySQL pour vous montrer comment cela fonctionne.

De plus, nous vous recommandons de jeter un oeil á `mylua`. Avec cela, vous pouvez utiliser le langage LUA pour charger une dynamiquement une procèdure dans `mysqld`.

9.3.1 La procèdure Analyse

`analyse([max elements,[max memory]])`

Cette procèdure est dèfinie dans le fichier '`sql/sql_analyse.cc`'. Elle examine les rèsultats de vos requêtes et en retourne une analyse :

- `max elements` (256 par dèfaut) est le nombre maximal de valeurs distinctes que `analyse` retiendra par colonne. C'est utilisè par `analyse` pour vèrifier si le type optimal de la colonne ne serait pas le type `ENUM`.
- `max memory` (8192 par dèfaut) est le maximum de mèmoire que `analyse` doit allouer par colonne quand elle essaye de trouver toutes les valeurs distinctes.

```
SELECT ... FROM ... WHERE ... PROCEDURE ANALYSE([max elements,[max memory]])
```

9.3.2 Ecrire une procèdure

Pour le moment, la seule documentation á ce sujet est constituèe par les sources.

Vous pourrez trouver toutes les informations sur les procèdures en examinant les fichiers suivants :

- '`sql/sql_analyse.cc`'
- '`sql/procedure.h`'
- '`sql/procedure.cc`'
- '`sql/sql_select.cc`'

Annexe A Problémes et erreurs communes

Ce chapitre liste quelques problémes et erreurs communes que les utilisateurs rencontreront. Vous apprendrez á dèterminer d'oú vient le probléme, et comment le rèsoudre. Vous trouverez aussi les solutions appropriées á quelques problémes communs.

A.1 Comment dèterminer ce qui pose probléme

Lorsque vous faite face á un probléme, la premiére chose á faire et de trouver quel programme / piéce de l'èquipement pose ce probléme :

- Si vous avez l'un des symptômes suivants, alors il est probable que cela soit un probléme matèriel (mèmoire, carte mére, processeur, ou disque dur) ou un probléme de noyau (kernel) :
  - Le clavier ne fonctionne pas. Cela peut être vèrifiè en pressant la touche Caps Lock (verrouillage majuscules). Si la lumiére de Caps Lock light ne change pas, vous devez remplacer votre clavier. (Avant de le faire, redèmarrez votre ordinateur aprés avoir vèrifiè les câbles du clavier.)
  - Le curseur de la souris ne bouge pas.
  - La machine ne rèpond pas á un ping externe.
  - D'autres programmes ne fonctionnent pas correctement.
  - Votre systéme a redèmarrè sans que vous vous y attendiez (un programme corrompu appartenant á un utilisateur ne devrait **jamais** être capable de couper votre systéme).

  Dans ce cas, vous devez commencer par vèrifier tout vos cables et dèmarrer quelques outils de diagnostic pour vèrifier votre matèriel ! Vous devez aussi regarder s'il existe des patches, mises á jours, ou packs de services pour votre systéme d'exploitation qui pourraient rèsoudre votre probléme. Vèrifiez aussi que vos librairies (comme glibc) sont á jour.

  Il est toujours bon d'utiliser une machine avec de la mèmoire ECC pour dècouvrir les problémes de mèmoire assez tôt !
- Si votre clavier est bloquè, vous pouvez rèparer cela en vous loggant sur votre machine á partir d'une autre machine et exècutant `kbd_mode -a`.
- Examinez votre fichier de log systéme (/var/log/messages ou similaire) pour connaître les raisons de vos problémes. Si vous pensez que le probléme vient de MySQL, vous devez aussi examiner les fichiers de log de MySQL. Voir Section 4.9.3 [Update log], page 328.
- Si vous ne pensez pas avoir de problémes au niveau du matèriel, vous devez trouver quel programme pose probléme.

  Essayez en utilisant `top`, `ps`, `taskmanager`, ou des programmes similaires, pour voir quel programme utilise trop de ressources ou bloque la machine.
- Vèrifiez avec `top`, `df`, ou un programme similaire si vous n'avez plus de mèmoire, d'espace disque, trop de fichiers ouverts ou un probléme avec une autre ressource critique.

- Si le probléme vient d'un processus, vous pouvez toujours essayer de le terminer. S'il ne veut pas se terminer, c'est probablement un bogue du systéme d'exploitation.

Si aprés tout cela vous pensez encore que le probléme vient du serveur MySQL ou du client MySQL, il est temps de prèparer un rapport de bogue pour notre liste de diffusion ou notre èquipe de support. Dans ce rapport, essayez de donner la description la plus dètaillèe possible du comportement du systéme et de ce que vous pensez qu'il se passe. Vous devez aussi mentionner pourquoi est-ce que vous pensez que le probléme vient de MySQL. Prenez en considèration toutes les situations dècrites dans ce chapitre. Dècrivez les problémes exactement comme ils surviennent sur votre systéme. Utilisez la mèthode 'copier/coller' pour les affichages et les messages d'erreurs provenant des programmes ou des fichiers de log.

Essayez de dècrire en dètail quel est le programme qui ne fonctionne pas et tous les symptômes que vous voyez ! Nous avons dèjá reçu beaucoup de rapports de bogue qui disaient juste "le systéme ne marche pas". Cela ne nous fournit aucune information á propos du probléme.

Si un programme èchoue, il est toujours utile de savoir :

- Est-ce que le programme en question a causè une erreur de segmentation (core dumped)?
- Est-ce que le programme consomme toutes les ressources CPU ? Vèrifiez avec `top` Laissez le programme fonctionne un bout de temps, il se peut qu'il soit entrain de traiter une tâche lourde.
- Si c'est le serveur `mysqld` qui pose probléme, pouvez vous essayer un `mysqladmin -u root ping` ou `mysqladmin -u root processlist` ?
- Que dit un programme client (essayez avec `mysql`, par exemple) quand vous essayez de vous connecter au serveur MySQL ? Le programme se bloque-t-il ? Obtenez vous un retour quelconque ?

Lors de l'envoi d'un rapport de bogue, vous devez respecter les régles dèfinies dans ce manuel. Voir Section 1.6.2.2 [Asking questions], page 27.

A.2 Erreurs communes rencontrèes avec MySQL

Cette section couvre les erreurs les plus frèquement rencontrèes par les utilisateurs. Vous trouverez ici une description de ces erreurs et un moyen de les corriger.

A.2.1 Erreur `Access denied`

Voir Section 4.2.6 [Privileges], page 209 et plus particuliérement Section 4.2.11 [Access denied], page 222.

A.2.2 Erreur `MySQL server has gone away`

Cette section couvre aussi l'erreur `Lost connection to server during query`.

Le plus souvent, l'erreur `MySQL server has gone away` se produit lorsque le serveur a dèpassè le dèlai d'attente d'actions et a clos la connexion. Par dèfaut, le serveur clos

la connexion aprés 8 heures si rien n'est arrivè. Vous pouvez changer cette limite en configurant le paramétre `wait_timeout` lorsque vous dèmarrez `mysqld`.

Une autre raison de recevoir l'erreur `MySQL server has gone away` est d'avoir emis un "close" sur votre connexion puis d'avoir essayè d'actionner une autre commande alors que la connexion ètait close.

Si vous avez un script, vous n'avez qu'á lancer la requête á nouveau pour que le client se reconnecte automatiquement.

Vous obtiendrez normalement les codes erreurs suivants dans ce cas (qui est indèpendant du systéme d'exploitation) :

Code erreur	**Description**
`CR_SERVER_GONE_ERROR`	Le client ne peut envoyer de commandes au serveur.
`CR_SERVER_LOST`	Le client n'a pas obtenu d'erreur en contactant le serveur, mais n'a pas obtenu de rèponse compléte á la question posèe.

vous obtiendrez aussi cette erreur si quelqu'un á terminè le processus avec `kill #idprocessus#`.

Vous pouvez vèrifier si le serveur MySQL est encore en marche en exècutant `mysqladmin version` et examinant la date de mise en route. Si le probléme est que mysqld a crashè, vous devriez vous concentrer sur la rèsolution du probléme. Vous devez dans ce cas commencer par vèrifier si emmétre la même requête fera á nouveau crasher MySQL. Voir Section A.4.1 [Crashing], page 697.

Vous pouvez aussi obtenir ces erreurs si vous envoyez une requête incorrecte ou trop grande au serveur. Si `mysqld` reçoit un paquet trop large ou mal ordonnè, il suppose que quelque chose s'est mal passè au niveau du client et ferme la connexion. Si vous avez besoin de grande requêtes (par exemple, si vous travaillez avec de grandes colonnes `BLOB`) vous pouvez augmenter la taille limite des requêtes en dèmarrant `mysqld` avec l'option `-O max_allowed_packet=#` (1M par dèfaut). Le surplus de mèmoire est allouè á la demande, ce qui fait que `mysqld` n'utilisera de la mèmoire que lorsque vous emmètrez une grande requête ou qu'il aura á retourner de grandes rèponses !

Si vous voulez rapporter un bogue concernant ce probléme, merci d'inclure les informations suivantes :

- MySQL a-t-il plantè ? (Vous pouvez le savoir en regardant le fichier `hostname.err`. Voir Section A.4.1 [Crashing], page 697.
- Si une requête spècifique fait planter `mysqld` et que les tables concernèes ont bien ètè vèrifièes avec `CHECK TABLE` avant l'exècution, pouvez-vous faire une batterie de tests ? Voir Section E.1.6 [Reproduceable test case], page 820.
- Quelle est la valeur de la variable `wait_timeout` dans le serveur MySQL ? `mysqladmin variables` vous donnera une rèponse
- Avez-vous essayè de dèmarrer `mysqld` avec `--log` et vèrifiè si la requête apparaît bien dans le log ?

Voir Section 1.6.2.2 [Asking questions], page 27.

A.2.3 Erreur `Can't connect to [local] MySQL server`

Un client MySQL sous Unix peut se connecter au serveur `mysqld` de deux façons diffèrentes : sockets Unix, qui se connectent via un fichier du systéme de fichiers ('`/tmp/mysqld.sock`' par dèfaut) ou TCP/IP, qui se connecte via un port. Les sockets Unix sont plus rapides que TCP/IP mais ne peuvent être utilisèe que pour des connexions locales. Les sockets sont utilisèes si vous ne spècifiez pas de nom d'hôte ou si vous spècifiez le nom d'hôte spècial `localhost`.

Sur Windows, si le serveur `mysqld` tourne sur 9x/Me, vous ne pouvez vous connecter qu'avec TCP/IP. Si le serveur tourne sur NT/2000/XP et que mysqld a ètè dèmarrè avec l'option `--enable-named-pipe`, vous pouvez aussi vous connectez avec un tunnel nommè. Son nom est MySQL. Si vous ne spècifiez pas un nom d'hôte lors de la connexion á `mysqld`, un client MySQL essayera d'abord de se connecter au tunnel nommè, et si cela ne marche pas il se connectera au port TCP/IP. Vous pouvez forcer l'utilisation des tunnels nommès sous Windows en utilisant `.` en tant que nom d'hôte.

L'erreur (2002) `Can't connect to ...` signifie gènèralement qu'il n'y a aucun serveur MySQL qui tourne sur la machine ou que vous utiliser un mauvais fichier de socket ou un port erronè quand vous essayez de vous connecter au serveur `mysqld`.

Commencez par vèrifier (en utilisant `ps` ou le gestionnaire de tâches sous Windows) qu'il y'a un processus nommè `mysqld` sur votre serveur ! S'il n'y'en a aucun, vous devrez en dèmarrer un. Voir Section 2.4.2 [Starting server], page 102.

Si un processus `mysqld` est actif, vous pouvez testerle serveur avec l'une des connexions suivantes (le port et le chemin vers la socket peuvent être diffèrents chez vous, bien sûr) :

```
shell> mysqladmin version
shell> mysqladmin variables
shell> mysqladmin -h 'hostname' version variables
shell> mysqladmin -h 'hostname' --port=3306 version
shell> mysqladmin -h 'ip de votre hôte' version
shell> mysqladmin --socket=/tmp/mysql.sock version
```

Notez l'utilisation des guillemets obliques plutôt que les simples avec la commande `hostname`; cela provoque la substitution de `hostname` par la valeur courante du nom d'hôte de la machine dans la commande `mysqladmin`.

Voilá quelques raisons pouvant entrainer l'erreur `Can't connect to local MySQL server` :

- `mysqld` ne fonctionne pas.
- Vous utilisez un systéme qui utilise les pthreads MIT. Si vous utilisez un systéme qui n'a pas le support natif des threads, `mysqld` utilises le package MIT-pthreads. Voir Section 2.2.3 [Which OS], page 74. Toutefois, toutes les versions de MIT-pthreads ne supportent pas les sockets Unix. Sur un systéme qui ne supporte pas les sockets vous devez toujours spècifier le nom d'hôte explicitement lors de la connexion au serveur. Utilisez cette commande pour vèrifier la connexion au serveur :

  ```
  shell> mysqladmin -h 'hostname' version
  ```

- Quelqu'un a effacè le fichier de socket Unix que `mysqld` utilise ('`/tmp/mysqld.sock`' par dèfaut). Vous avez peut-être une tâche `cron` qui efface la socket MySQL (par exemple, une tâche qui supprime les anciens fichiers du dossier '`/tmp`'). Vous pouvez

toujours exècuter `mysqladmin version` et vèrifier que la socket que `mysqladmin` tente d'utiliser existe vraiment. La solution dans ce cas est de modifier la tâche `cron` pour qu'elle n'efface plus '`mysqld.sock`' ou de placer la socket quelque part d'autre. Voir Section A.4.5 [Problems with mysql.sock], page 702.

- Vous avez dèmarrè `mysqld` avec l'option `--socket=/chemin/vers/socket`. Si vous changez le chemin vers la socket vous devez aussi en notifier les clients. Vous pouvez le faire en fournissant le chemin vers la socket en argument au client. Voir Section A.4.5 [Problems with mysql.sock], page 702.
- Vous utilisez Linux et un thread s'est terminè (core dumped). Dans ce cas, vous devez aussi terminer les autres threads `mysqld` (par exemple, avec le script `mysql_zap` avant de pouvoir dèmarrer un nouveau serveur MySQL. Voir Section A.4.1 [Crashing], page 697.
- Vous n'avez peut-être pas les priviléges de lecture et ècriture sur le dossier contenant la socket ou sur la socket elle-même. Dans ce cas, vous devez changer les droits sur ce dossier / fichier ou redèmarrer `mysqld` pour qu'il prenne en compte un dossier auquel vous avec accés.

Si vous obtenez l'erreur `Can't connect to MySQL server on un_hôte`, vous pouvez essayer ce qui suit pour trouver le probléme :

- Vèrifiez que le serveur fonctionne en faisant `telnet votre-nom-d-hôte port-tcp-ip` et pressez la touche Enter plusieurs fois. Si il y a un serveur MySQL qui tourne sur ce port, vous devriez obtenir une rèponse contenant le numèro de version du serveur. Si vous obtenez une erreur proche de `telnet: Unable to connect to remote host: Connection refused`, c'est qu'il n'y a pas de serveur tournant sur le port donnè.
- Essayez de vous connecter au dèmon `mysqld` sur la machine locale et vèrifiez le port TCP/IP de la configuration de `mysqld` (variable `port`) avec `mysqladmin variables`.
- Vèrifiez que votre serveur `mysqld` n'est pas configurè avec l'option `--skip-networking`.

A.2.4 `Host '...' is blocked` Error

Si vous obtenez cette erreur :

```
Host 'hostname' is blocked because of many connection errors.
Unblock with 'mysqladmin flush-hosts'
```

cela veut dire que `mysqld` a reçu trop de (`max_connect_errors`) tentatives de connexions á l'hôte `'hostname'` qui ont ètè interrompus en plein milieu. Aprés `max_connect_errors` requêtes èchouèes, `mysqld` pense qu'il se passe quelque chose de mauvais (comme une attaque de la part d'un pirate), et bloque le serveur pour les prochaines connexions jusqu'á ce que quelqu'un exècute la commande `mysqladmin flush-hosts`.

Par dèfaut, `mysqld` bloque le serveur aprés 10 connexions erronèes. Vous pouvez facilement changer ce comportement en dèmarrant le serveur avec ces arguments :

```
shell> safe_mysqld -O max_connect_errors=10000 &
```

Notez que si vous recevez ce message pour un hôte en particulier, vous devriez vous assurer qu'il n'y a pas de problémes de connexions TCP/IP depuis cet hôte. Si vos connexions TCP/IP ne marchent pas, il ne servira á rien d'augmenter la valeur de la variable `max_connect_errors`!

A.2.5 Erreur `Too many connections`

Si vous obtenez l'erreur `Too many connections` en essayant de vous connecter á MySQL, cela signifie qu'il y'a dèjá `max_connections` clients connectès au serveur `mysqld`.

Si vous avez besoin de plus de connexion que par dèfaut (100), vous devez redèmarrer `mysqld` avec une plus grande valeur pour la variable `max_connections`.

Notez que `mysqld` permet actuellement á (`max_connections+1`) clients de se connecter. La derniére connexion est rèservèe á l'utilisateur ayant le privilége `SUPER`. En ne donnant pas ce privilége aux utilisateurs normaux (ils ne devraient pas en avoir besoin), un administrateur avec ce privilége peut se connecter et utiliser `SHOW PROCESSLIST` pour trouver ce qui pose probléme. Voir Section 4.5.6 [SHOW], page 267.

Le nombre maximal de connexion MySQL dèpend de la qualitè de la librairie des threads sur une plate-forme donnèe. Linux et Solaris devraient être capables de supporter jusqu'á 500-1000 connexion simultanèes, cela dèpend èvidemment de la quantitè de RAM que vous avez et de ce que font les clients.

A.2.6 Erreur `Some non-transactional changed tables couldn't be rolled back`

Si vous obtenez l'erreur / avertissement : `Warning: Some non-transactional changed tables couldn't be rolled back` en essayant de faire un `ROLLBACK`, cela signifie que certaines tables que vous avez utilisè dans la transaction ne supportent pas les transactions. Ces tables non-transactionneles ne seront pas affectèes par la commande `ROLLBACK`.

Le cas le plus typique pour que cela arrive est que vous avez essayè de crèer une table d'nu type non supportè par votre binaire `mysqld`. Si `mysqld` ne supporte pas un type de table (ou si le type de table est dèsactivè par une option de dèmarrage), il crèera une table avec le type qui se rapproche le plus de celui que vous aviez choisi, probablement `MyISAM`.

Vous pouvez vèrifier le type d'une table en faisant :

`SHOW TABLE STATUS LIKE 'nom_de_table'`. Voir Section 4.5.6.2 [SHOW TABLE STATUS], page 269.

Vous pouvez vèrifier les extensions que votre binaire `mysqld` supporte en faisant :

`show variables like 'have_%'`. Voir Section 4.5.6.4 [SHOW VARIABLES], page 273.

A.2.7 Erreur `Out of memory`

Si vous lancez une requête et que vous obtenez l'erreur suivante :

```
mysql: Out of memory at line 42, 'malloc.c'
mysql: needed 8136 byte (8k), memory in use: 12481367 bytes (12189k)
ERROR 2008: MySQL client ran out of memory
```

notez que cette erreur parle du client MySQL client `mysql`. La raison de cette erreur est simplememnt que le client n'a pas la mèmoire suffisante pour stocker le rèsultat en entier.

Pour remèdier á ce probléme, vèrifiez d'abord que votre requête est correcte. Est-ce normal qu'elle retourne autant de lignes ? Si oui, vous pouvez utiliser `mysql --quick`, qui utilise `mysql_use_result()` pour rècupèrer les rèsultats. Cela solicittera moins le client (mais plus le serveur).

A.2.8 Erreur `Packet too large`

Lorsqu'un client MySQL ou le serveur `mysqld` reçoit un paquet plus grand que `max_allowed_packet` octets, il provoque une erreur `Packet too large` et ferme la connexion.

En MySQL 3.23 le plus gros paquet possible est 16M (á cause des limites du protocole client/serveur). En MySQL 4.0.1 et plus, cela n'est plus limitè que par la quantitè de mèmoire que vous avez sur votre serveur (cela va thèoriquement á un maximum de 2G).

Un paquet de communication est une simple commande SQL envoyèe au serveur, ou une simple ligne renvoyèe au client.

Lorsqu'un client MySQL ou que le serveur `mysqld` reçoit un paquet plus grand que `max_allowed_packet` octets, il provoque une erreur `Packet too large` et ferme la connexion. avec quelques clients, vous pouvez aussi obtenir l'erreur `Lost connection to MySQL server during query` si le paquet est trop grand.

Notez que le client et le serveur ont chacun leur propre variable `max_allowed_packet`. Si vous voulez gèrer les gros paquets, vous devrez changer cette variable côtè client et côtè serveur.

Il n'est pas dangereux d'augmenter cette valeur ètant donnè que la mèmoire n'est allouè que lorsque besoin en est. Cette variable est plus une prècaution pour capturer les paquets erronès qui circulent entre le client et le serveur. Elle sert aussi a vous assurer que vous n'utilisez pas accidentellement de gros paquets qui consommeront toute la mèmoire.

Si vous utilisez le client `mysql`, vous pouvez spècifier un plus grand tampon en dèmarrant le client avec `mysql --set-variable=max_allowed_packet=8M`. Les autres clients ont diffèrentes mèthodes pour configurer cette variable. Notez que `--set-variable` est dèsapprouvèe depuis MySQL 4.0, utilisez `--max-allowed-packet=8M` á la place.

Vous pouvez utiliser le fichier d'options pour augmenter la taille de `max_allowed_packet` dans `mysqld`. Par exemple, si vous vous attendez á stocker la totalitè d'un `MEDIUMBLOB` dans une table, vous aurez besoin de dèmarrer le serveur avec l'option `set-variable=max_allowed_packet=16M`.

Vous pouvez aussi rencontrer d'ètranges problémes avec les gros paquets si vous utilisez les grands blob, mais que vous n'avez pas donnè á `mysqld` l'accés á assez de mèmoire pour gèrer ces requêtes. Si vous pensez être dans ce cas, essayez d'ajouter `ulimit -d 256000` au dèbut du script `safe_mysqld` et redèmarrez `mysqld`.

A.2.9 Erreurs de communication / Connexion annulèe

A partir de `MySQL 3.23.40` vous n'obtenez l'erreur `Aborted connection` que si vous dèmarrez `mysqld` avec `--warnings`.

Si vous trouvez des erreurs comme celle qui suit dans vos logs d'erreurs :

```
010301 14:38:23  Aborted connection 854 to db: 'users' user: 'josh'
```

Voir Section 4.9.1 [Error log], page 328.

Cela signifie qu'un probléme est survenu :

- Le programme client n'a pas appelè `mysql_close()` avant de quitter.
- Le client a ètè inactif plus de `wait_timeout` ou `interactive_timeout` secondes. Voir Section 4.5.6.4 [SHOW VARIABLES], page 273.

- L'exècution du programme client s'est terminèe soudainement au milieu d'un transfert.

Lorsque ce qui prècéde arrive, la variable `Aborted_clients` est incrèmentèe.

La variable serveur `Aborted_connects` est incrèmentèe lorsque :

- Un paquet de connexion ne contient pas la bonne information.
- L'utilisateur n'avait pas les droits d'accés á la base de donnèes.
- L'utilisateur a utilisè un mot de passe erronè.
- Il a fallu plus de `connect_timeout` secondes pour obtenir un paquet de communication.

Notez que ce qui prècéde peut indiquer que quelqu'un essaye de s'introduire dans votre base de donnèes !

Voir Section 4.5.6.4 [SHOW VARIABLES], page 273.

Autres raisons pour les problémes de clients echouès / connexions interrompues :

- L'utilisation du Duplex du protocole Ethernet, qu'il soit intègral (Full-Duplex) ou partiel (Half Duplex) avec Linux. La plupart des pilotes Ethernet de Linux ont ce bogue. Vous pouvez tester ce bogue en transfèrant un ènorme fichier via FTP d'une machine á l'autre. Si le transfert est saccadè (succession de transfert-pause-transfert-pause) alors vous vivez le syndrôme du duplex de Linux. La seule solution á ce probléme est de dèsactiver á la fois le duplex partiel et le duplex intègral sur les concentrateurs et les commutateurs.
- Quelques problémes avec la librairie des threads qui causent des interruptions de lectures.
- Mauvaise configuration TCP/IP.
- Les câbles, concentrateurs ou commutateurs Ethernet dèffectueux. On peut le diagnostiquer aisèment en remplacement le matèriel.
- `max_allowed_packet` est trop petit ou les requêtes ont besoin de plus de mèmoire que celle que vous avez allouè á `mysqld`. Voir Section A.2.8 [Packet too large], page 690.

A.2.10 Erreur `The table is full`

Il y'a diffèrents cas oú vous pouvez obtenir cette erreur :

- Vous utilisez une ancienne version de MySQL (avant 3.23.0) quand une table temporaire en mèmoire devient plus grande que `tmp_table_size` octets. Pour èviter ce probléme, vous pouvez utiliser l'option `-O tmp_table_size=#` pour faire augmenter la taille des tables temporaires á `mysqld` ou utiliser l'option SQL `BIG_TABLES` avant d'exècuter la requête qui pose probléme. Voir Section 5.5.6 [SET], page 396.

  Vous pouvez aussi dèmarrer `mysqld` avec l'option `--big-tables`. Cela revient á utiliser `BIG_TABLES` pour toutes les requêtes.

  Dans la version 3.23 de MySQL, les tables temporaires en mèmoire seront automatiquement changèes en tables physique `MyISAM` aprés qu'elles n'aient dèpassè `tmp_table_size`.
- Vous utilisez des tables `InnoDB` et avez dèpassè leur taille. Dans ce cas, la solution est d'augmenter les tailles des tables.

- Vous utilisez des tables `ISAM` ou `MyISAM` sur un systéme d'exploitation qui ne supporte pas les fichiers de plus de 2G et vous avez atteint cette limite dans le fichier de donnèes ou d'index.
- Vous utilisez des tables `MyISAM` et la taille des donnèes ou de l'index est plus grande que celle que MySQL a allouè aux pointeurs. (Si vous ne spècifiez pas `MAX_ROWS` á `CREATE TABLE` MySQL n'allouera que des pointeurs supportant 4G de donnèes).

  Vous pouvez obtenir la taille maximale des donnèes / index en faisant :

  ```
  SHOW TABLE STATUS FROM database LIKE 'nom_de_table';
  ```

  or using `myisamchk -dv database/nom_de_table`.

  Si le probléme vient de lá, vous pouvez le corriger en faisant quelque chose se rapprochant de :

  ```
  ALTER TABLE nom_de_table MAX_ROWS=1000000000 AVG_ROW_LENGTH=nnn;
  ```

  Vous n'avez besoin de spècifier `AVG_ROW_LENGTH` que pour les tables avec des champs BLOB/TEXT car dans ce cas, MySQL ne peut optimiser l'espace requis en se basant uniquement sur le nombre de lignes.

A.2.11 Erreur `Can't create/write to file`

Si vous obtenez une erreur de ce type pour quelques requêtes :

```
Can't create/write to file '\\sqla3fe_0.ism'.
```

cela signifie que MySQL ne peut crèer de fichier temporaire pour le jeu de rèsultats dans le dossier temporaire dèfini. (L'erreur prècèdente est typique de Windows, et le message d'erreur Unix est similaire.) La solution est de dèmarrer `mysqld` avec l'option `--tmpdir=chemin` ou d'ajouter á votre fichier d'options :

```
[mysqld]
tmpdir=C:/temp
```

en supposant que le dossier '`c:\\temp`' existe. Voir Section 4.1.2 [Option files], page 198.

Vèrifiez aussi le code erreur que vous obtenez avec `perror`. Une autre rasion peut être une erreur de disque saturè.

```
shell> perror 28
Error code  28:  No space left on device
```

A.2.12 Erreur du client `Commands out of sync`

Si vous obtenez l'erreur `Commands out of sync; you can't run this command now`, le probléme vient du fait que vous appelez les fonctions dans le mauvais ordre dans votre code !

Cela peut se produire, par exemple, si vous utilisez `mysql_use_result()` et essayez d'exècuter une nouvelle requête avant d'avoir appelè `mysql_free_result()`. Cela peut aussi se produire si vous essayez d'exècuter deux requêtes qui retournent des donnèes dans appeler `mysql_use_result()` ou `mysql_store_result()` entre les deux.

A.2.13 Erreur `Ignoring user`

Si vous obtenez l'erreur suivante :

`Found wrong password for user: 'some_user@some_host'; ignoring user`

cela signifie que lors du dèmarrage de `mysqld` ou lorsqu'il a rechargè les tables de permissions, il a trouvè une entrèe dans la table `user` avec un mot de passe invalide. De ce fait, l'entrèe est tout simplement ignorèe par le systéme de droits.

Causes possibles et solutions pour ce probléme :

- Vous faites peut-être tourner une nouvelle version de `mysqld` avec une vielle table `user`. Vous pouvez vèrifier cela en exècutant `mysqlshow mysql user` pour voir si le champ du mot de passe est plus petit que 16 caractéres. Si c'est le cas, vous pouvez le corriger en exècutant le script `scripts/add_long_password`.
- L'utilisateur a un ancien mot de passe (8 caractéres) et vous n'avez pas dèmarrè `mysqld` avec l'option `--old-protocol`. Mettez á jour le mot de passe dans la table `user` ou redèmarrez `mysqld` avec `--old-protocol`.
- Vous avez spècifiè un mot de passe dans la table `user` sans passer par la fonction `PASSWORD()`. Utilisez `mysql` pour mettre á jour l'utilisateur dans la table `user` avec un nouveau mot de passe. Assurez-vous d'utiliser la fonction `PASSWORD()` :

```
mysql> UPDATE user SET password=PASSWORD('votre mot de passe')
    ->             WHERE user='XXX';
```

A.2.14 Erreur `Table 'xxx' doesn't exist`

Si vous obtenez l'erreur `Table 'xxx' doesn't exist` ou `Can't find file: 'xxx' (errno: 2)`, cela signifie qu'aucune table portant le nom `xxx` n'existe dans la base de donnèes courante.

Notez que comme MySQL utilise des dossiers et des fichiers pour stocker les bases de donnèes et les tables, les noms sont **sensibles á la casse** ! (Sous Windows les noms ne sont pas sensibles á la casse, mais vous devez utiliser la même casse dans une même requête !)

Vous pouvez obtenir la liste des tables que vous avez dans la base de donnèes courante avec la commande `SHOW TABLES`. Voir Section 4.5.6 [SHOW], page 267.

A.2.15 Erreur `Can't initialize character set xxx`

Si vous obtenez l'erreur suivante :

```
MySQL Connection Failed: Can't initialize character set xxx
```

Cela signifie l'une des choses suivantes :

- Le jeu de caractéres est un jeu multi-octets et votre client ne les supporte pas.

  Dans ce cas, vous devez recompiler le client avec `--with-charset=xxx` ou avec `--with-extra-charsets=xxx`. Voir Section 2.3.3 [configure options], page 88.

  Tous les binaires standards de MySQL sont compilès avec `--with-extra-character-sets=complex` qui active le support de tous les jeux de caractéres multi-octets. Voir Section 4.6.1 [Character sets], page 286.

- Le jeu de caractéres est un simple jeu de caractéres qui n'est pas compilè dans `mysqld` et les fichiers de dèfinition du jeu ne sont pas á l'endroit oú le client si attend.

  Dans ce cas vous avez besoin de :

  - Recompiler le client avec le support du jeu de caractéres. Voir Section 2.3.3 [configure options], page 88.
  - Spècifier au client oú les fichiers de dèfinition du jeu de caractéres se situent. Pour beaucoup de clients, vous pouvez le faire avec l'option `--character-sets-dir=chemin-vers-dossier-jeu-caractéres`.
  - Copier les fichiers de dèfinition du jeu de caractéres dans le dossier oú le client s'attend á les trouver.

A.2.16 Fichier non trouvè

Si vous obtenez `ERROR '...' not found (errno: 23)`, `Can't open file: ... (errno: 24)`, ou toute autre erreur avec `errno` 23 ou `errno` 24 de la part de MySQL, cela signifie que vous n'avez pas allouè assez de descripteurs de fichiers á MySQL. Vous pouvez utiliser l'utilitaire `perror` pour obtenir une description de ce que veut dire l'identifiant de l'erreur :

```
shell> perror 23
File table overflow
shell> perror 24
Too many open files
shell> perror 11
Resource temporarily unavailable
```

Le probléme ici est que `mysqld` essaye de garder trop de fichiers ouverts en même temps. Vous pouvez soit demander á `mysqld` de ne pas ouvrir autant de fichiers simultanèment ou augmenter le nombre de descripteurs de fichiers allouès á `mysqld`.

Pour dire á `mysqld` de garder moins de fichiers ouverts en même temps, vous pouvez rendre le cache de tables plus petit en utilisant l'option `-O table_cache=32` de `safe_mysqld` (la valeur par dèfaut est 64). Rèduire la valeur de `max_connections` rèduira aussi le nombre de fichiers ouverts (90 comme valeur de dèfaut).

Pour changer le nombre de descripteurs de fichiers allouès á `mysqld`, vous pouvez utiliser l'option `--open-files-limit=#` de `safe_mysqld` ou `-O open-files-limit=#` de `mysqld`. Voir Section 4.5.6.4 [SHOW VARIABLES], page 273. La façon la plus facile de faire cela est d'ajouter cette option dans votre fichiers d'options. Voir Section 4.1.2 [Option files], page 198. Si vous avec une ancienne version de `mysqld` qui ne le supporte pas, vous pouvez èditer le script `safe_mysqld`. Il y'a une ligne commentèe `ulimit -n 256` dans le script. Vous pouvez enlever le caractére `'#'` pour dècommenter cette ligne, et changer le nombre 256 pour affecter le nombre de descripteurs de fichiers allouès á `mysqld`.

`ulimit` (et `open-files-limit`) peuvent augmenter le nombre de descripteurs de fichiers, mais seulement jusqu'á la limite imposèe par le systéme d'exploitation. Il y'a aussi une limite 'matèrielle' qui ne peut être dèpassèe que si vous dèmarrez `safe_mysqld` ou `mysqld` en tant que root (souvenez-vous juste que vous devez aussi utiliser l'option `--user=...` dans ce cas). Si vous avez besoin de repousser les limites du systéme d'exploitation pour les descripteurs de fichiers disponibles pour chaque processus, consultez la documentation de votre systéme.

Notez que si vous dèmarrez le shell `tcsh`, `ulimit` ne fonctionnera pas ! `tcsh` retournera aussi des valeurs incorrectes si vous atteignez la limite courante ! Dans ce cas, vous devez dèmarrer `safe_mysqld` avec `sh` !

A.3 Notes relatives á l'installation

A.3.1 Problémes lors de la liaison avec la librairie du client MySQL

Si vous liez votre programme et que vous obtenez des erreurs pour des symboles non-rèfèrencès qui commencent par `mysql_`, comme ce qui suit :

```
/tmp/ccFKsdPa.o: In function ‘main’:
/tmp/ccFKsdPa.o(.text+0xb): undefined reference to ‘mysql_init’
/tmp/ccFKsdPa.o(.text+0x31): undefined reference to ‘mysql_real_connect’
/tmp/ccFKsdPa.o(.text+0x57): undefined reference to ‘mysql_real_connect’
/tmp/ccFKsdPa.o(.text+0x69): undefined reference to ‘mysql_error’
/tmp/ccFKsdPa.o(.text+0x9a): undefined reference to ‘mysql_close’
```

vous pouvez rèparer cela en ajoutant `-Lchemin-vers-la-librairie-mysql -lmysqlclient` **last** dans votre ligne de liaison.

Si vous obtenez une erreur `undefined reference` pour la fonction `uncompress` ou `compress`, ajoutez `-lz` **á la fin** de votre ligne de liaison et essayez á nouveau !

Si vous obtenez des erreurs `undefined reference` pour des fonctions qui devraient exister sur votre systéme, comme `connect`, vèrifiez la page de manuel de la fonction en question, pour les librairies que vous devez ajouter á la ligne de liaison !

Si vous obtenez une erreur `undefined reference` pour des fonctions inexistantes sur votre systéme, ressemblant á ce qui suit :

```
mf_format.o(.text+0x201): undefined reference to ‘__lxstat’
```

cela signifie que votre librairie est compilè sur un systéme qui n'est pas á 100% compatible avec le votre. Dans ce cas, vous devez obtenir la derniére distribution des sources de MySQL et compiler vous-mêmes. Voir Section 2.3 [Installing source], page 84.

Si vous essayez de faire fonctionner un programme et que vous obtenez des erreurs pour des symboles non-rèfèrencès qui commencent par `mysql_` ou une erreur disant que la librairie `mysqlclient` ne peut être trouvèe, cela signifie que votre systéme n'arrive pas á trouver la librairie partagèe '`libmysqlclient.so`'.

La solution est de dire á votre systéme de chercher les librairies partagèes lá oú la librairie est situèe avec l'une des mèthodes suivantes :

- Ajouter le chemin vers le dossier oú se situe '`libmysqlclient.so`' dans la variable d'environnement `LD_LIBRARY_PATH`.
- Ajouter le chemin vers le dossier oú se situe '`libmysqlclient.so`' dans la variable d'environnement `LD_LIBRARY`.
- Copiez le fichier '`libmysqlclient.so`' á un endroit oú votre systéme le cherche, comme dans le dossier '`/lib`', et mettez á jour les informations de la librairie partagèe en exècutant `ldconfig`.

Un autre moyen de rèsoudre ce probléme est de lier votre programme statiquement, avec `-static`, ou en effaçant les librairies dynamiques de MySQL avant de lier votre code. Dans le second cas vous devez vous assurer qu'aucun autre programme n'utilise les librairies dynamiques !

A.3.2 Comment exècuter MySQL comme un utilisateur normal

Le serveur MySQL `mysqld` peut être dèmarrè par n'importe quel utilisateur. Afin de changer l'utilisateur qui fait tourner `mysqld` en l'utilisateur Unix `nom_utilisateur`, vous devez faire ceci :

1. Stoppez le serveur si il fonctionne (utilisez `mysqladmin shutdown`).
2. Changez le propriètaire du dossier et des fichiers de bases pour qu'il soit `nom_utilisateur`. Il faut que cet utilisateur ait les droits d'ècriture et de lecture (vous pourriez avoir á faire cette manipulation en tant que `root` Unix) :

   ```
   shell> chown -R nom_utilisateur /path/to/mysql/datadir
   ```

   Si les dossier ou les fichiers de donnèes de MySQL sont des liens symboliques, vous devez vous assurer de pouvoir suivre ces lignes, et de changer les propriètaires des fichiers et dossiers sur lesquels ils pointent. L'option `chown -R` de chown peut ne pas suivre les liens symboliques.
3. Dèmarrez le serveur avec l'utilisateur `nom_utilisateur`, ou bien, si vous utilisez MySQL version 3.22 ou plus rècent, dèmarrez `mysqld` en tant que `root` Unix, et utilisez l'option `--user=nom_utilisateur`. `mysqld` va alors changer automatiquement d'utilisateur pour utiliser `nom_utilisateur` avant d'accepter les connexions.
4. Pour dèmarrer le serveur sous le nom d'utilisateur automatiquement au moment du dèmarrage du systéme, ajouter une ligne `user` qui spècifie le nom de l'utilisateur que le groupe de `[mysqld]` est du même groupe que le fichier d'options '`/etc/my.cnf`' ou le fichier d'options '`my.cnf`' dans le dossier de donnèes du serveur. Par exemple :

   ```
   [mysqld]
   user=nom_utilisateur
   ```

A ce moment, votre processus `mysqld` doit fonctionner normalement sous le nom de l'utilisateur Unix `nom_utilisateur`. Une chose n'a pas changè : les droits dans les tables de droits de MySQL. Par dèfaut, juste aprés avoir exècutè le script d'installation des tables de droits `mysql_install_db`, l'utilisateur MySQL `root` est le seul utilisateur du systéme avec les droits de crèer et de dètruire les bases. A moins que vous n'ayez changè ces droits, ils ont toujours cours. Cela ne va pas vous empêcher d'accèder á MySQL en tant que `root` MySQL, même si vous n'êtes pas connectè en tant que `root` Unix. Spècifiez simplement l'option `-u root` au programme client.

Notez qu'accèder á MySQL en tant que `root`, en fournissant l'option `-u root` en ligne de commande, **n'a rien a voir** avec MySQL qui fonctionne avec les droits de `root` Unix, ou d'un autre utilisateur Unix. Les droits d'accés et les noms d'utilisateurs MySQL sont complétement sèparè des noms d'utilisateurs et des mots de passes Unix. Le seul rapport avec les utilisateurs Unix est que si vous ne fournissez pas l'option `-u` lorsque vous dèmarrez votre client, le client va essayer de se connecter á MySQL avec votre nom d'utilisateur Unix.

Si votre serveur Unix n'est pas sècurisè, il est recommandè de donner un mot de passe á l'utilisateur MySQL `root` dans les tables de droits. Sinon, n'importe quel utilisateur ayant

un compte sur cette machine va pouvoir accèder au compte root avec l'option `mysql -u root nom_base` et faire ce qu'il veut.

A.3.3 Problémes avec les permissions sur fichiers

Si vous avez des problémes avec les droits sur fichiers, par exemple, si `mysql` gènére l'erreur suivante lorsque vous crèez une table :

```
ERROR: Can't find file: 'path/with/filename.frm' (Errcode: 13)
```

alors la variable d'environnement `UMASK` est peut-être mal configurèe lorsque `mysqld` est dèmarrè. La valeur par dèfaut de umask est `0660`. Vous pouvez corriger ce comportement en dèmarrant `safe_mysqld` de la façon suivante :

```
shell> UMASK=384  # = 600 en octal
shell> export UMASK
shell> /chemin/vers/safe_mysqld &
```

Par dèfaut, MySQL crèera les dossiers des bases de donnèes et de `RAID` avec 0700 comme type de permissions. Vous pouvez modifier ce comportement en dèfinissant la variable `UMASK_DIR`. Si vous le faite, les nouveaux dossiers seront crèes en combinant `UMASK` et `UMASK_DIR`. Par exemple, si vous voulez donner un accés de groupe á tout les nouveaux dossiers, vous pouvez faire :

```
shell> UMASK_DIR=504  # = 770 en octal
shell> export UMASK_DIR
shell> /chemin/vers/safe_mysqld &
```

A partir de la version 3.23.25, MySQL suppose que les valeurs de `UMASK` et `UMASK_DIR` sont en octal si elles commencent par un zèro.

Voir Annexe F [Environment variables], page 828.

A.4 Notes relatives á l'administration

A.4.1 Que faire si MySQL crashe constamment ?

Toutes les versions de MySQL sont testèes sur plusieurs plate-formes avant leur publication. Cela ne signifie pas qu'elles sont exemptèes de bogues, cela signifie juste que si il y'a des bogues, il y'en a trés peu et sont durs á trouver. Si vous avez un probléme, cela nous aidera toujours si vous essayez de trouver d'oú vient exactement le crashage systéme, et vous aurez plus de chances de le voir rèsolu rapidement.

D'abord, vous devez essayer de trouver si le probléme vient du dèmon `mysqld` qui se termine, ou s'il est liè á votre client. Vous pouvez savoir depuis combien de temps le serveur `mysqld` tourne en exècutant `mysqladmin version`. Si `mysqld` s'est terminè, vous trouverez sûrement la raison dans le fichier '`mysql-data-directory/‘hostname‘.err`'. Voir Section 4.9.1 [Error log], page 328.

Plusieurs crashes de MySQL sont causès par des fichiers de donnèes ou d'index corrompus. MySQL ècrira les donnèes sur le disque avec un appel systéme á `write()`, aprés chaque

requête et avant d'en notifier le client. (Cela n'est pas vrai si vous utilisez `delay_key_write`, auquel cas seul les donnèes sont ècrites.) Cela signifie que les donnèes sont intégres même si `mysqld` crashe, puisque le systéme d'exploitation s'assurera que les donnèes non sorties du tampon ne sont pas enregistrèe sur le disque. Vous pouvez forcer MySQL á se synchroniser avec le disque aprés chaque requête en dèmarrant `mysqld` avec `--flush`.

Ce qui prècéde signifie que normalement, vous ne devriez obtenir de tables corrompues que si :

- Quelqu'un ou quelque chose a coupè `mysqld` ou la machine au milieu d'une mise á jour.
- Vous avez trouvè un bogue dans `mysqld` qui le termine au milieu d'une mise á jour.
- Quelqu'un manipule les fichiers de donnèes ou d'index en dehors de **mysqld** sans verrouiller proprement les tables.
- Si vous faites tourner plusieurs serveurs `mysqld` avec les mêmes donnèes sur un systéme qui ne gére pas bien les vèrrous de fichiers (normalement gèrès par le dèmon `lockd`) ou que vous le faites avec `--skip-external-locking`
- Vous avez un fichier de donnèes ou d'index corrompu qui contient des donnèes faussèes ce qui améne `mysqld` á confusion.
- Vous avez trouvè un bogue dans le systéme de stockage des donnèes. Cela parait impossible, mais sait-on jamais ? Dans ce cas, essayez de changer le type de fichier pour qu'il soit pris en charge par un autre gestionnaire de bases de donnèes en utilisant `ALTER TABLE` sur une copie rèparèe de la table !

Parce qu'il est trés difficile de savoir pourquoi quelque chose crashe, essayez d'abord de voir si les choses qui marchent pour les autres ne fonctionnent pas chez vous. Merci d'essayer les diffèrentes choses suivantes :

Coupez le dèmon `mysqld` avec `mysqladmin shutdown`, exècutez `myisamchk --silent --force */*.MYI` sur toutes les tables, et redèmarrez le dèmon `mysqld`. Cela vous assurera que vous partez d'un bon point de dèpart. Voir Chapitre 4 [MySQL Database Administration], page 192.

- Utilisez `mysqld --log` et essayez de dèterminer á partir des informations du log si une requête spècifique fait planter le serveur. Plus de 95% de tous les bogues sont liès á une requête spècifique ! Normalement, c'est la derniére requête dans le fichier de log avant que MySQL n'ait redèmarrè. Voir Section 4.9.2 [Query log], page 328. Si vous pouvez faire planter MySQL á plusieurs reprise avec une requête, même aprés avoir vèrifiè toutes les tables avant de l'exècuter, alors vous avez trouvè le bogue et vous devez faire un rapport de bogue pour nous en avertir ! Voir Section 1.6.2.3 [Bug reports], page 27.
- Essayer d'effectuer une batterie de tests que nous pourrons utiliser pour reproduire le probléme. Voir Section E.1.6 [Reproduceable test case], page 820.
- Essayez d'exècuter le test inclus mysql-test et les benchmarks MySQL. Voir Section 9.1.2 [MySQL test suite], page 670. Ils devraient tester plutôt bien MySQL. Vous pouvez aussi ajouter ce code au benchmarks pour simuler votre application ! Les benchmarks peuvent être trouvès dans le rèpertoire '`bench`' dans la distribution des sources ou, pour une distribution binaire, dans le rèpertoire '`sql-bench`' de votre dossier d'installation MySQL.
- Essayez `fork_test.pl` et `fork2_test.pl`.

- Si vous configurez MySQL pour le dèbogage, il sera plus facile d'obtenir des informations á propos des erreurs possibles si quelque chose se passe mal. Reconfigurez MySQL avec l'option `--with-debug` ou `--with-debug=full` de `configure` puis recompilez. Voir Section E.1 [Debugging server], page 815.
- Configurer MySQL pour le dèbogage inclu un outil d'allocation de mèmoire qui peut trouver quelques erreurs. Il fournit aussi beaucoup d'informations sur ce qui se passe.
- Avez-vous appliquè les derniers patches de votre systéme d'exploitation ?
- Utilisez l'option `--skip-external-locking` de `mysqld`. Sur quelques systémes, le gestionnaire des verrous `lockd` ne fonctionne pas convenablement; l'option `--skip-external-locking` dit á `mysqld` de ne pas utiliser de pas utiliser de verrous externes. (Cela signifie que vous ne pouvez pas faire tourner deux serveurs `mysqld` sur les mêmes donnèes et que vous devez faire attention si vous utilisez `myisamchk`, mais il peut être instructif d'essayer cette option comme test.)
- Avez-vous essayè `mysqladmin -u root processlist` lorsque `mysqld` semble fonctionner mais ne rèpond plus ? Quelquefois, `mysqld` n'est pas comateux, même si vous le croyez. Le probléme peut-être que toutes les connexions sont utilisèes, ou qu'il y'a quelques problémes avec les verrous internes. `mysqladmin processlist` devra normalement être en mesure d'effectuer une connexion même dans ce cas, et peut fournir des informations utiles á propos du nombre de connexions courantes et de leur status.
- Exècutez la commande `mysqladmin -i 5 status` ou `mysqladmin -i 5 -r status` ou dans une fenêtre sèparèe pour produire des statistiques pendant que vous exècutez vos autres requêtes.
- Essayez ce qui suit :
  1. Dèmarrez `mysqld` á partir de `gdb` (ou d'un autre dèbogueur). Voir Section E.1.3 [Using gdb on mysqld], page 817.
  2. Exècutez vos scripts de tests.
  3. Affichez le traçage et les variables locales aux trois niveaux les plus bas. Avec gdb vous pouvez le faire avec les commandes suivantes lorsque `mysqld` a crashè á l'intèrieur de gdb :

     ```
     backtrace
     info local
     up
     info local
     up
     info local
     ```

     Avec gdb vous pouvez aussi savoir quels threads existent avec `info threads` et en prendre un avec `thread #`, oú `#` est l'identifiant du thread.
- Essayez de simuler votre application avec un script Perl pour forcer MySQL á crasher ou á avoir un comportement dèfectueux.
- Envoyez un rapport de bogue normal. Voir Section 1.6.2.3 [Bug reports], page 27. Soyez le plus prècis possible et donnez plus de dètails que d'habitude. Puisque MySQL fonctionne pour beaucoup de personnes, il se peut que le crash rèsulte de quelque chose de spècifique á votre systéme (par exemple, une erreur liée á la particularitè de vos librairies systéme).

- Si vous avez des problémes avec des tables á lignes de longueurs dynamiques et que vous n'utilisez pas de colonnes `BLOB/TEXT` (mais seulement des colonnes `VARCHAR`), vous pouvez essayer de changer tous les `VARCHAR` en `CHAR` avec `ALTER TABLE`. Cela forcera MySQL á utiliser des lignes de tailles fixes. Les lignes á tailles fixèes prennent un peu plus d'espace, mais sont plus tolèrants aux corruptions !

  Le code courant des lignes dynamiques est utilisè chez MySQL AB depuis au moins 3 ans sans aucun probléme, mais par nature, les lignes á longueur dynamique sont plus exposèes aux erreurs, il est donc bon d'essayer ce qui prècéde pour voir si cela vous aide !

A.4.2 Comment rèinitialiser un mot de passe Root oubliè

Si vous n'avez jamais configurè un mot de passe `root` pour MySQL, le serveur n'en demendera jamais un pour toutes les connexions de cet utilisateur. Il est recommandè de toujours assigner un mot de passe á chaque utilisateur. Voir Section 4.2.2 [Security], page 206.

Si vous avez configurè un mot de passe pour l'utilisateur `root`, mais que vous l'avez oubliè, vous pouvez en choisir un nouveau en suivant la procèdure suivante :

1. Terminez le serveur `mysqld` en lui envoyant un `kill` (pas un `kill -9`). L'identifiant du processus est stockè dans un fichier '`.pid`', qui est normalement situè dans le dossier des bases de donnèes de MySQL :

   ```
   shell> kill `cat /dossier-donnees-mysql/hote.pid`
   ```

   Vous devez être l'utiliateur Unix `root` ou l'utilisateur qui fait tourner `mysqld` pour pouvoir le faire.
2. Redèmarrez `mysqld` avec l'option `--skip-grant-tables`.
3. Choisissez un nouveau mot de passe avec la commande `mysqladmin password` :

   ```
   shell> mysqladmin -u root password 'nouveaumotdepasse'
   ```
4. Maintenant, vous pouvez soit stopper `mysqld` et le redèmarrer normalement, ou bien juste charger les tables de priviléges avec :

   ```
   shell> mysqladmin -h hote flush-privileges
   ```
5. Aprés cela, vous devriez pouvoir vous connecter avec le nouveau mot de passe.

Vous pouvez aussi choisir le nouveau mot de passe en utilisant le client `mysql` :

1. Stoppez et redèmarrez `mysqld` avec l'option `--skip-grant-tables` comme dècrit plus haut.
2. Connectez vous au serveur `mysqld` avec :

   ```
   shell> mysql -u root mysql
   ```
3. Exècutez la commande suivante dans le client `mysql` :

   ```
   mysql> UPDATE user SET Password=PASSWORD('nouveaumotdepasse')
       ->             WHERE User='root';
   mysql> FLUSH PRIVILEGES;
   ```
4. Aprés cela, vous devriez pouvoir vous connecter avec le nouveau mot de passe.
5. Vous pouvez maintenant stopper et redèmarrer `mysqld` de façon normale.

A.4.3 Comment MySQL gére un disque plein

Lorsqu'il n'y a plus d'espace disque, MySQL fait ce qui suit :

- Il vèrifie chaque minute pour voir s'il y'a assez d'espace pour ècrire la ligne courante. Si oui, il continue comme si rien ne s'ètait passè.
- Chaque 6 minutes, il ajoute un avertissement dans le fichier de log á propos de la condition du disque.

Pour contourner ce probléme, vous pouvez effectuer les actions suivantes :

- Pour continuer, il suffit juste d'avoir assez d'espace disque pour insèrer tous les enregistrements.
- Pour annuler le thread, vous devez lui envoyer un `mysqladmin kill`. Le thread sera annulè la prochaine fois qu'il vèrifiera le disque (dans 1 minute).
- Notez que d'autres threads peuvent être en train d'attendre pour accèder á la table qui a causè le probléme de disque plein. Si vous avez beaucoup de threads "locked", terminer le thread qui a causè la dèfaillance permettra aux autres threads de continuer.

Les exceptions pour le comportement suivant sont lorsque vous utilisez `REPAIR` ou `OPTIMIZE` ou lorsque les index sont crèès dans un batch aprés l'exècution de `LOAD DATA INFILE` ou d'un `ALTER TABLE`.

Toutes les commandes prècèdentes risquent d'utiliser de gros fichiers temporaires, qui pourraient perturber le reste du systéme s'ils n'ètaient pas supprimès. Si MySQL obtient une erreur de disque plein lors de l'exècution d'une des commandes prècèdentes, il effacera les gros fichiers temporaires et marquera la table comme crashèe (á part pour `ALTER TABLE`, oú l'ancienne table sera restaurèe).

A.4.4 Oú MySQL stocke les fichiers temporaires ?

MySQL utilise la valeur de la variable d'environnement `TMPDIR` comme chemin du dossier oú stocker les fichiers temporaires. Si vous n'avez pas de variable `TMPDIR`, MySQL utilise alors le dossier systéme par dèfaut, qui est normalement '`/tmp`' ou '`/usr/tmp`'. Si le support qui contient votre dossier temporaire est trop petit, modifiez le script `safe_mysqld` pour configurer `TMPDIR`, et lui faire dèsigner un dossier oú vous aurez la place. Vous pouvez aussi configurer ce dossier avec l'option `--tmpdir` de `mysqld`.

MySQL crèe tous les fichiers temporaires sous forme de fichier cachès. Cela garantit que les fichiers temporaires seront supprimès lorsque `mysqld` est terminè. L'inconvènient d'utiliser les fichiers cachès est que vous ne verrez pas que le dossier temporaire est gros, et qu'ils risquent de remplir votre dossier temporaire.

Lors des tris avec les clauses `ORDER BY` ou `GROUP BY`, MySQL utilise normalement deux dossiers temporaires. L'espace disque maximal nècessaire est :

```
(taille de ce qui est trià + taille du pointeur de base)
* nombre de lignes trouvèes
* 2
```

`taille du pointeur de base` vaut gènèralement 4, mais peut croître dans le futur pour les tables rèellement grandes.

Pour certaines requêtes `SELECT`, MySQL crèe aussi des tables temporaires SQL. Elles ne sont pas cachèes, et portent un nom du type '`SQL_*`'.

`ALTER TABLE` crèe une table temporaire dans le même dossier que la table originale.

A.4.5 Comment protèger ou changer le fichier socket '/tmp/mysql.sock'

Si vous avez un probléme avec le fait que n'importe qui peut effacer le fichier de socket de communication MySQL '`/tmp/mysql.sock`', vous pouvez, dans la plupart des versions d'Unix, protèger votre systéme de fichiers '`/tmp`' en lui assignant le bit `sticky`. Connectez vous en `root` et faite ce qui suit :

```
shell> chmod +t /tmp
```

Cela protègera votre systéme de fichiers '`/tmp`' de sorte que les fichiers ne pourront être effacès que par leurs propriètaires ou le super utilisateur (`root`).

Vous pouvez vèrifier que le bit `sticky` est actif en exècutant `ls -ld /tmp`. Si le dernier bit de permission est `t`, il l'est.

Vous pouvez changer l'endroit oú MySQL utilise / place le fichier de socket de la façon suivante :

- Spècifiez le chemin dans un fichier d'options globales ou locales. Par exemple, placez dans `/etc/my.cnf` :

  ```
  [client]
  socket=chemin-vers-fichier-socket

  [mysqld]
  socket=chemin-vers-fichier-socket
  ```

  Voir Section 4.1.2 [Option files], page 198.
- Spècifiez cela en ligne de commande á `safe_mysqld` et á la plupart des clients avec l'option `--socket=chemin-vers-fichier-socket`.
- Spècifiez le chemin vers la socket dans la variable d'environnement `MYSQL_UNIX_PORT`.
- Dèfinissez le chemin avec l'option de `configure --with-unix-socket-path=chemin-vers-fichier-socket`. Voir Section 2.3.3 [configure options], page 88.

Vous pouvez vèrifier que la socket fonctionne avec cette commande :

```
shell> mysqladmin --socket=/chemin/vers/socket version
```

A.4.6 Problémes de fuseaux horaires

Si vous avez un probléme avec `SELECT NOW()` qui retournerait des valeurs en GMT et non votre temps local, vous devez configurer la variable d'environnement `TZ` et la mettre sur votre fuseau horaire courant. Cela peut être fait pour l'environnement dans lequel le serveur fonctionne, par exemple, dans `safe_mysqld` ou `mysql.server`. Voir Annexe F [Environment variables], page 828.

A.5 Problémes relatifs aux requêtes

A.5.1 Sensibilitè á la casse dans les recherches

Par dèfaut, les recherchez de MySQL ne sont pas sensibles á la casse (cependant, il existe des jeux de caractéres qui ne sont jamais insensibles á la casse, comme `czech`).

Cela signifie que si vous recherchez avec `nom_colonne LIKE 'a%'`, vous aurez toutes les valeurs de la colonne qui commencent par un `A` ou un `a`. Si vous voulez que cette recherche soit sensible á la casse, utilisez par exemple `INSTR(nom_colonne, "A")=1` pour vèrifier un prèfixe. Utilisez `STRCMP(nom_colonne, "A") = 0` si la valeur de la colonne doit être exactement `"A"`.

Les opèrations de comparaisons simples (`>=, >, = , < , <=`, tri et groupement) sont basèes sur la "valeur de tri" de chaque caractére. Les caractéres avec la même valeur de tri (comme E, e et è) sont considèrès comme le même caractére !

Dans les anciennes versions de MySQL les comparaisons avec `LIKE` ètaient effectuèes sur la majuscule de chaque caractére (E == e mais E <> è). Dans les nouvelles versions `LIKE` fonctionne comme les autres opèrateurs de comparaison.

Si vous voulez qu'une colonne soit toujours traitèe de façon sensible á la casse, dèclarez lá en tant que `BINARY`. Voir Section 6.5.3 [`CREATE TABLE`], page 504.

Si vous utilisez des donneès chinoises avec l'encodage big5, vous devez rendre toutes les colonnes de chaînes `BINARY`. Cela fonctionne car l'ordre de tri de l'encodage big5 est basè sur l'ordre des codes ASCII.

A.5.2 Problémes avec l'utilisation des colonnes `DATE`

Le format d'une valeur de `DATE` est `'YYYY-MM-DD'`. En accord avec ANSI SQL, aucun autre format n'est autorisè. Vous devez utiliser ce format dans les `UPDATE` et les clauses `WHERE` des requêtes `SELECT`. Par exemple :

```
mysql> SELECT * FROM nom_de_table WHERE date >= '1997-05-05';
```

MySQL convertit automatiquement une date en nombre si la date est utilisèe dans un contexte numèrique (et vice versa). Il est aussi assez intelligent pour permettre une forme "relaxèe" lors des mises á jour et dans les clauses `WHERE` qui comparent une date et une colonne `TIMESTAMP`, `DATE`, ou `DATETIME`. (Forme relaxèe signifie que n'importe quel caractére de ponctuation peut être utilisè en tant que sèparateurs des parties. Par exemple, `'1998-08-15'` et `'1998#08#15'` sont èquivalents.) MySQL peut convertir une chaîne ne contenant aucun sèparateur (comme `'19980815'`), en supposant qu'elle a un sens pour une date.

La date spèciale `'0000-00-00'` peut être stockèe et rècupèrèe en tant que `'0000-00-00'`. Lors de l'utilisation d'une date `'0000-00-00'` avec `MyODBC`, elle sera automatiquement convertie en `NULL` á partir de la version 2.50.12 de `MyODBC`, car ODBC ne peut gèrer ce type de dates.

Puisque MySQL effectue les conversions dècrites plus haut, ce qui suit fonctionnera :

```
mysql> INSERT INTO nom_de_table (idate) VALUES (19970505);
mysql> INSERT INTO nom_de_table (idate) VALUES ('19970505');
```

```
mysql> INSERT INTO nom_de_table (idate) VALUES ('97-05-05');
mysql> INSERT INTO nom_de_table (idate) VALUES ('1997.05.05');
mysql> INSERT INTO nom_de_table (idate) VALUES ('1997 05 05');
mysql> INSERT INTO nom_de_table (idate) VALUES ('0000-00-00');

mysql> SELECT idate FROM nom_de_table WHERE idate >= '1997-05-05';
mysql> SELECT idate FROM nom_de_table WHERE idate >= 19970505;
mysql> SELECT MOD(idate,100) FROM nom_de_table WHERE idate >= 19970505;
mysql> SELECT idate FROM nom_de_table WHERE idate >= '19970505';
```

Toutefois, ce qui suit ne fonctionnera pas :

```
mysql> SELECT idate FROM nom_de_table WHERE STRCMP(idate,'19970505')=0;
```

`STRCMP()` est une fonction de chaînes de caractéres, il convertit donc `idate` en une chaîne et effectue une comparaison de chaînes. Il ne convertit pas `'19970505'` en date et n'effectue donc pas de comparaison de dates.

Notez que MySQL vèrifie trés peu l'intègritè des dates. Si vous stockez une date erronèe, comme `'1998-2-31'`, la date erronèe sera enregistrèe.

Vu que MySQL compresse les dates pour les stocker, il ne peut stocker tout format donnè car il risquerait de ne pas correspondre au tampon de rèsultat. Les régles d'acceptations de dates sont :

- Si MySQL peut enregistrer et rècupèrer une date donnèe, la date erronèe est acceptèe pour les colonnes `DATE` et `DATETIME`
- Toutes les valeurs de ours comprises entre 0 et 31 sont acceptèes. Cela est fort convenable pour les applications web oú vous demandez l'annèe, mois et jour dans trois champs textes (ou liste dèroulantes) diffèrents.
- Le champ jour ou mois peut être un zèro. Cela est convenable si vous voulez enregistrer un anniversaire dans une colonne `DATE` et que vous ne connaissez qu'une partie de la date.

Si la date ne peut être convertie en une valeur raisonnable, un `0` est insèrè dans le champ `DATE`, il sera rècupèrè en tant que `0000-00-00`. Cela est une solution rapide et convenue vu que nous considèrons que la responsabilitè de la base de donnèes est de rècupèrer la même date que vous aviez stockè (même si la date n'est pas correcte). Nous pensons que c'est á l'application de vèrifier les dates, et non au serveur de le faire.

A.5.3 Problémes avec les valeurs `NULL`

Le concept de la valeur `NULL` est une source de confusions pour les dèbutants en SQL, qui pensent souvent que `NULL` est la même chose qu'une chaîne de caractéres vide `""`. Ce n'est pas le cas ! Par exemple, les deux requêtes suivantes sont complétement diffèrentes :

```
mysql> INSERT INTO ma_table (telephone) VALUES (NULL);
mysql> INSERT INTO ma_table (telephone) VALUES ("");
```

Les deux requêtes insérent des valeurs dans la colonne `telephone`, mais la premiére insére une valeur `NULL` et la seconde insére une chaîne vide. La signification de la premiére peut être “le numèro de tèlèphone est inconnu” et la seconde peut être considèrèe comme “elle n'a pas de tèlèphone”.

En SQL, la valeur NULL est toujours false en comparaison á n'importe quelle autre valeur, même NULL. Une expression contenant NULL produit toujours un rèsultat NULL sauf si une indication contraire est prèsente dans la documentation des opèrateurs et des fonctions impliquèes dans l'expression. Toutes les colonnes de l'exemple suivant retournent NULL :

```
mysql> SELECT NULL,1+NULL,CONCAT('Invisible',NULL);
```

Si vous voulez trouver les colonnes dont la valeur est NULL, vous ne pouvez pas utiliser le test =NULL. La requête suivante ne retourne aucune ligne car expr = NULL est FALSE, pour n'importe quelle expression :

```
mysql> SELECT * FROM ma_table WHERE telephone = NULL;
```

Pour trouver les valeurs NULL, vous devez utiliser le test IS NULL. Ce qui suit montre comment trouver les numèros de tèlèphone NULL et les numèros vides :

```
mysql> SELECT * FROM ma_table WHERE telephone IS NULL;
mysql> SELECT * FROM ma_table WHERE telephone = "";
```

Notez que vous ne pouvez ajouter d'index qu'aux colonnes pouvant avoir la valeur NULL si vous utilisez la version 3.23.2 de MySQL ou plus rècente avec des tables de type MyISAM ou InnoDB. Dans les versions prècèdentes et avec les autres types, vous devez dèclarer de telles colonnes NOT NULL. Cela signifie aussi que vous ne pouvez pas insèrer NULL dans les colonnes indexèes.

Lors de la lecture de donnèes avec LOAD DATA INFILE, les colonnes vides sont interprètèes en tant que ''. Si vous voulez une valeur NULL dans une colonne, vous devez utiliser \N dans le fichier. Le mot littèral 'NULL' peut aussi être utilisè dans certaines circonstances. Voir Section 6.4.9 [LOAD DATA], page 497.

Lors de l'utilisation de ORDER BY, les valeurs NULL sont prèsentèes en premier. Si vous triez dans l'ordre dècroissant en utilisant DESC, les valeurs NULL sont prèsentèes en dernier. Lors de l'utilisation de GROUP BY, toutes les valeurs NULL sont considèrèes comme ègales.

Pour mieux gèrer les valeurs NULL, vous pouvez utiliser les opèrateurs IS NULL et IS NOT NULL et la fonction IFNULL().

Pour certains types de colonnes, les valeurs NULL sont traitèes spècialement. Si vous insèrez NULL dans la premiére colonne TIMESTAMP d'une table, la date et le temps courants sont insèrès. Si vous insèrez NULL dans une colonne AUTO_INCREMENT, le nombre suivant de la sèquence sera insèrè.

A.5.4 Problémes avec les alias

Vous pouvez utiliser un alias pour vous rèfèrer á une colonne dans une clause GROUP BY, ORDER BY, ou HAVING. Les alias peuvent aussi être utilisès pour donner de meilleurs noms aux colonnes :

```
SELECT SQRT(a*b) as rt FROM nom_de_table GROUP BY rt HAVING rt > 0;
SELECT id,COUNT(*) AS cnt FROM nom_de_table GROUP BY id HAVING cnt > 0;
SELECT id AS "Identitè du client" FROM nom_de_table;
```

Notez que ANSI SQL ne vous permet pas de vous rèfèrer á un alias dans une clause WHERE. Il en est ainsi car lorsque le code de WHERE est exècutè, la valeur de la colonne ne peut pas encore être dèterminèe. Par exemple, la requête suivante est **illègale** :

```
SELECT id,COUNT(*) AS cnt FROM nom_de_table WHERE cnt > 0 GROUP BY id;
```

La clause `WHERE` est exècutèe pour savoir quelles lignes devraient être inclues dans la partie `GROUP BY` tandis que `HAVING` est utilisè pour dècider quelles lignes du jeu de rèsultats doivent être utilisèes.

A.5.5 Effacer des lignes de tables reliées

Comme MySQL ne supporte encore ni les sous-requêtes, ni l'utilisation de plusieurs tables dans une requête `DELETE` (avant la version 4.0), vous devez utiliser l'approche suivante pour effacer des lignes de deux tables reliées :

1. Sèlectionnez (`SELECT`) les lignes de la table principale en vous basant sur une condition `WHERE`.
2. Effacez (`DELETE`) les lignes de la table principale en vous basant sur la même condition.
3. `DELETE FROM table_liée WHERE colonne_liée IN (lignes_sèlèctionnèes)`.

Si le nombre total des caractéres dans la requête avec la `colonne_liée` est supèrieur á 1,048,576 (la valeur par dèfaut est `max_allowed_packet`, vous devez la dècouper en parties plus petites et exècuter plusieurs `DELETE`. Vous obtiendrez probablement les suppressions les plus rapides en n'effaçant que 100-1000 `colonne_liée` par requête si `colonne_liée` est un index. Si ce n'est pas un index, la vitesse est indèpendante du nombre d'arguments dans la clause `IN`.

A.5.6 Rèsoudre les problémes des lignes non retournèes

Si vous avez une requête complexe avec beaucoup de tables et qu'elle ne retourne aucun rèsultat, vous devez suivre la procèdure suivante pour trouver ce qui cloche dans votre requête :

1. Testez la requête avec `EXPLAIN` et vèrifiez si vous trouvez quelque chose qui vous parait fausse. Voir Section 5.2.1 [`EXPLAIN`], page 363.
2. Ne sèlèctionnez que les champs que vous utilisez dans la clause `WHERE` :
3. Enlevez une table á la fois de la requête jusqu'á ce qu'elle retourne quelques lignes. Si les tables sont grandes, il est bon d'utiliser `LIMIT 10` dans la requête.
4. Exècutez un `SELECT` pour les colonnes qui auraient du trouver des lignes dans la derniére table supprimèe de la requête.
5. Si vous comparez des colonnes `FLOAT` ou `DOUBLE` avec des nombres á virgule, vous ne pouvez pas utiliser `'='`. C'est un probléme commun á la plupart des langages de programmation car les valeurs á virgules flottantes ne sont pas des valeurs exactes. Dans le plupart des cas, changer la colonnes `FLOAT` en `DOUBLE` corrigera cela. Voir Section A.5.7 [Problems with float], page 707.
6. si vous ne pouvez toujours pas trouver ce qui ne va pas, crèez un test minimal pouvant être exècutè avec `mysql test < query.sql` montrant votre probléme. Vous pouvez crèer un fichier de test avec `mysqldump --quick base tables > query.sql`. Editez le fichier et supprimez quelques lignes d'insertions (s'il y'en a trop), et ajoutez votre requête de sèlèction á la fin du fichier.

   Vèrifiez que vous avez encore le probléme en faisant :

```
shell> mysqladmin create test2
shell> mysql test2 < query.sql
```

Envoyez le fichier de test, en utilisant `mysqlbug`, á `mysql@lists.mysql.com`.

A.5.7 Problémes de comparaisons avec nombres á virgule flottante

Les nombres á virgule flottante portent souvent á confusion, car ils ne sont pas enregistrès en tant que valeurs exactes dans l'architecture de l'ordinateur. Ce qu'on voit á l'ècran n'est souvent pas la valeur exacte du nombre.

Ce discours s'applique aux champs de types `FLOAT`, `DOUBLE` et `DECIMAL`.

```
CREATE TABLE t1 (i INT, d1 DECIMAL(9,2), d2 DECIMAL(9,2));
INSERT INTO t1 VALUES (1, 101.40, 21.40), (1, -80.00, 0.00),
(2, 0.00, 0.00), (2, -13.20, 0.00), (2, 59.60, 46.40),
(2, 30.40, 30.40), (3, 37.00, 7.40), (3, -29.60, 0.00),
(4, 60.00, 15.40), (4, -10.60, 0.00), (4, -34.00, 0.00),
(5, 33.00, 0.00), (5, -25.80, 0.00), (5, 0.00, 7.20),
(6, 0.00, 0.00), (6, -51.40, 0.00);

mysql> SELECT i, SUM(d1) AS a, SUM(d2) AS b
    -> FROM t1 GROUP BY i HAVING a <> b;
+------+--------+-------+
| i    | a      | b     |
+------+--------+-------+
|    1 |  21.40 | 21.40 |
|    2 |  76.80 | 76.80 |
|    3 |   7.40 |  7.40 |
|    4 |  15.40 | 15.40 |
|    5 |   7.20 |  7.20 |
|    6 | -51.40 |  0.00 |
+------+--------+-------+
```

Le rèsultat est correcte. Même si les cinq premiers enregistrements ne devraient pas passer le test á priori, ils le font sûrement car la diffèrence entre les nombres se situe plus loin que les dècimales, ou que cela dèpend de l'architecture de l'ordinateur.

Le probléme ne peut être rèsolu en utilisant ROUND() (ou une fonction similaire), car le rèsultat est encore un nombre á virgule flottante. Exemple :

```
mysql> SELECT i, ROUND(SUM(d1), 2) AS a, ROUND(SUM(d2), 2) AS b
    -> FROM t1 GROUP BY i HAVING a <> b;
+------+--------+-------+
| i    | a      | b     |
+------+--------+-------+
|    1 |  21.40 | 21.40 |
|    2 |  76.80 | 76.80 |
|    3 |   7.40 |  7.40 |
|    4 |  15.40 | 15.40 |
|    5 |   7.20 |  7.20 |
```

```
|    6 | -51.40 |  0.00 |
+------+--------+-------+
```

Voilá ce á quoi les nombres dans le champ 'a' ressemblent :

```
mysql> SELECT i, ROUND(SUM(d1), 2)*1.0000000000000000 AS a,
    -> ROUND(SUM(d2), 2) AS b FROM t1 GROUP BY i HAVING a <> b;
+------+----------------------+-------+
| i    | a                    | b     |
+------+----------------------+-------+
|    1 |  21.3999999999999986 | 21.40 |
|    2 |  76.7999999999999972 | 76.80 |
|    3 |   7.4000000000000004 |  7.40 |
|    4 |  15.4000000000000004 | 15.40 |
|    5 |   7.2000000000000002 |  7.20 |
|    6 | -51.3999999999999986 |  0.00 |
+------+----------------------+-------+
```

Selon l'architecture de votre ordinateur, vous pouvez obtenir ou non les mêmes rèsultats. Chaque CPU peut èvaluer les nombres á virgule flottante d'une maniére diffèrente. Par exemple, sur des machines vous pouvez obtenir les rèsultats 'correctes' en multipliant les deux arguments par 1, l'exemple suivant illustre cela.

ATTENTION : NE FAITES JAMAIS CONFIANCE A CETTE METHODE DANS VOS APPLICATIONS, C'EST UN EXEMPLE DE MAUVAISE METHODES !!!

```
mysql> SELECT i, ROUND(SUM(d1), 2)*1 AS a, ROUND(SUM(d2), 2)*1 AS b
    -> FROM t1 GROUP BY i HAVING a <> b;
+------+--------+------+
| i    | a      | b    |
+------+--------+------+
|    6 | -51.40 | 0.00 |
+------+--------+------+
```

La raison pour laquelle l'exemple prècèdent semble fonctionner est que sur la machine en particulier oú le test a ètè effectuè, l'arithmètique des nombres á virgule flottante du CPU arrondi les nombres á la même valeur, mais il n'y a aucune régle stipulant qu'un CPU doit faire cela, on ne peut donc pas s'y fier.

La façon correcte d'effectuer des comparaisons de nombre á virgule flottante est d'en premier lieu dècider du degrè de tolèrance voulu entre les nombres puis d'effectuer la comparaison selon le nombre de tolèrance. Par exemple, si nous dècidons que les nombres á virgule flottante sont considèrès comme ègaux si ils ont la même prècision au dix-milliéme prés (0.0001), la comparaison ressemblera á cela :

```
mysql> SELECT i, SUM(d1) AS a, SUM(d2) AS b FROM t1
    -> GROUP BY i HAVING ABS(a - b) > 0.0001;
+------+--------+------+
| i    | a      | b    |
+------+--------+------+
|    6 | -51.40 | 0.00 |
+------+--------+------+
1 row in set (0.00 sec)
```

Et vice versa, si vous voulez obtenir les lignes oú les nombres sont les mêmes, le test sera :

```
mysql> SELECT i, SUM(d1) AS a, SUM(d2) AS b FROM t1
    -> GROUP BY i HAVING ABS(a - b) < 0.0001;
+------+-------+-------+
| i    | a     | b     |
+------+-------+-------+
|    1 | 21.40 | 21.40 |
|    2 | 76.80 | 76.80 |
|    3 |  7.40 |  7.40 |
|    4 | 15.40 | 15.40 |
|    5 |  7.20 |  7.20 |
+------+-------+-------+
```

A.6 Questions relatives aux dèfinitions de tables

A.6.1 Problémes avec ALTER TABLE.

`ALTER TABLE` change une table avec le jeu de caractéres courant. Si durant l'exècution d'`ALTER TABLE` vous obtenez une erreur de clef dupliquèe, alors la cause est soit que le nouveau jeu de caractéres interpréte deux clefs á la même valeur ou que la table est corrompue, au quel cas vous devez exècuter `REPAIR TABLE` sur la table.

Si `ALTER TABLE` se termine avec une erreur de ce genre :

```
Error on rename of './database/name.frm' to './database/B-a.frm' (Errcode: 17)
```

le probléme est peut-être que MySQL a crashè lors d'un prècèdent appel á `ALTER TABLE` et qu'il y a une ancienne table nommèe '`A-quelquechose`' ou '`B-quelquechose`' qui subsiste. Dans ce cas, dèplacez-vous dans le dossier des donnèes MySQL et effacez tout les fichiers dont les noms commencent par `A-` ou `B-`. (Vous voudrez peut-être les dèplacer autre part plutôt que de les effacer.)

`ALTER TABLE` fonctionne de la façon suivante :

- Crèe une nouvelle table nommèe '`A-xxx`' avec les changements voulus.
- Toutes les lignes de l'ancienne table sont copièes dans '`A-xxx`'.
- L'ancienne table est renommèe '`B-xxx`'.
- '`A-xxx`' est renommèe avec le nom de votre ancienne table.
- '`B-xxx`' est supprimèe.

Si quelque chose se passe mal durant l'opèration de changement de nom, MySQL essaye d'annuler les changements. Si quelque chose de grave se passe (cela ne devrait jamais arriver bien sûr), MySQL peut laisser l'ancienne table en tant que '`B-xxx`', mais un simple renommage au niveau systéme devrait restaurer vos donnèes.

A.6.2 Comment changer l'ordre des colonnes dans une table

Le but de SQL est de sèparer l'application du format de stockage des donnèes. Vous devriez toujours spècifier l'ordre dans lequel vous voulez rècupèrer vos donnèes. Par exemple :

```
SELECT nom_colonne1, nom_colonne2, nom_colonne3 FROM nom_de_table;
```

retournera les colonnes dans l'ordre `nom_colonne1`, `nom_colonne2`, `nom_colonne3`, tandis que :

```
SELECT nom_colonne1, nom_colonne3, nom_colonne2 FROM nom_de_table;
```

retournera les colonnes dans l'ordre `nom_colonne1`, `nom_colonne3`, `nom_colonne2`.

Vous ne devez **jamais**, dans une application, utiliser `SELECT *` et rècupèrer l'ordre des colonnes en vous basant sur leurs positions, car l'ordre dans lequel elles sont retournèes ne peut-être garantit tout le temps. Une simple modification dans votre base de donnèes peut endommager gravement vos applications.

Si vous voulez changer l'ordre de vos colonnes, vous pouvez faire ce qui suit :

1. Crèer une nouvelle table avec les colonnes dans le bon ordre.
2. Exècuter `INSERT INTO nouvelle_table SELECT champs-dans-nouvel-ordre FROM ancienne_table`.
3. Effacer ou renommer `ancienne_table`.
4. `ALTER TABLE nouvelle_table RENAME ancienne_table`.

A.6.3 Problémes avec les tables temporaires

Voici une liste des limitations avec `TEMPORARY TABLES`.

- Une table temporaire ne peut être que du type `HEAP`, `ISAM`, `MyISAM` ou `InnoDB`.
- Vous ne pouvez utiliser une table plus d'une fois dans une requête. Par exemple, ce qui suit ne marche pas :

  ```
  mysql> SELECT * FROM temporary_table, temporary_table AS t2;
  ```

  Nous allons corriger ce qui prècéde dans la version 4.0.
- Vous ne pouvez utiliser `RENAME` sur une table `TEMPORARY`. Notez que `ALTER TABLE nom_origine RENAME nouveau_nom` fonctionne !

  Nous allons corriger ce qui prècéde dans la version 4.0.

Annexe B Contributions

De nombreux utilisateurs de MySQL ont contribuès á ces utilitaires *trés* pratiques.

Une liste est disponible sur `http://www.mysql.com/Downloads/Contrib/` (ou sur un mirroir).

Visitez notre portail de logiciels á `http://www.mysql.com/portal/software/`. Ce site communautaire facilite vos propres contributions!

Si vous voulez compiler MySQL avec les interfaces Perl `DBI/DBD`, vous devriez commencer par tèlècharger les fichiers de `Data-Dumper`, `DBI` et `Msql-Mysql-modules`, et les installer. Voir Section 2.7 [Perl support], page 149.

Note : les programmes listès ci-dessous peuvent être librement tèlèchargès et utilisès. Ils portent le copyright de leurs propriètaires respectifs. Reportez-vous aux documentations de chacun de ces programme pour avoir plus de dètails sur les licences. MySQL AB n'assume aucune responsabilitè quant á l'exactitude des informations de cette page, ou de l'utilisation approprièe des logiciels qui y sont listès.

B.1 APIs

- Perl Modules
  - `http://www.mysql.com/Downloads/Contrib/Data-Dumper-2.101.tar.gz` Module Perl `Data-Dumper`. Utilie avec `DBI/DBD` pour le support des anciennes installations Perl.
  - `http://www.mysql.com/Downloads/Contrib/DBI-1.18.tar.gz` Module Perl `DBI`.
  - `http://www.mysql.com/Downloads/Contrib/KAMXbase1.2.tar.gz` Conversion de fichier '`.dbf`' en tables MySQL et rèciproquement. Le module Perl a ètè ècrit par Pratap Pereira `pereira@ee.eng.ohio-state.edu`, et amèliorè par Kevin A. McGrail `kmcgrail@digital1.peregrinehw.com`. Ce convertisseur supporte les champs `MEMO`.
  - `http://www.mysql.com/Downloads/Contrib/Msql-Mysql-modules-1.2218.tar.gz` Module Perl `DBD` pour accèder aux bases mSQL et MySQL.
  - `http://www.mysql.com/Downloads/Contrib/Data-ShowTable-3.3.tar.gz` Module Perl `Data-ShowTable`. Utile pour le support de `DBI/DBD`.
  - `http://www.mysql.com/Downloads/Contrib/HandySQL-1.1.tar.gz` HandySQL est un module d'accés á MySQL. Il offre une interface en C, intègrèe á Perl, qui est 20% plus rapide que DBI.
- OLEDB
  - `http://www.mysql.com/Downloads/Win32/MyOLEDB3.exe` Installeur MyOLEDB 3.0 par SWSoft.
  - `http://www.mysql.com/Downloads/Win32/mysql-oledb-3.0.0.zip` Source de MyOLEDB 3.0.
  - `http://www.mysql.com/Downloads/Win32/MySamples.zip` Exemples et documentation de MyOLEDB.

  - http://www.mysql.com/Downloads/Win32/MyOLEDB.chm Fichiers d'aide de MyOLEDB.
  - http://www.mysql.com/Downloads/Win32/libmyodbc.zip Librairie MyODBC statique, utilisèe pour compiler MyOLEDB. Basèe sur du code MyODBC.
- C++
  - http://www.mysql.com/Downloads/Contrib/mysql-c++-0.02.tar.gz Librairie MySQL C++. Par Roland Haenel, rh@ginster.net.
  - http://www.mysql.com/Downloads/Contrib/MyDAO.tar.gz API MySQL C++. Par Satish spitfire@pn3.vsnl.net.in. Inspirè de la librairie de Roland Haenel's C++ et de la librairie MyC de Ed Carp.
  - http://www.mysql.com/download_mysql++.html API MySQL C++ (plus qu'une simple couche supplèmentaire). Concept original de kevina@clark.net. Aujourd'hui gèrè par Sinisa á MySQL AB.
  - http://nelsonjr.homepage.com/NJrAPI/ Une couche d'abastraction de base de donnèes C++ qui supporte MySQL.
- Delphi
  - http://www.mysql.com/Downloads/Contrib/DelphiMySQL2.zip Interface Delphi avec libmysql.dll, par bsilva@umesd.k12.or.us.
  - http://www.mysql.com/Downloads/Contrib/Udmysql.pas Une interface pour libmysql.dll, á utiliser avec Delphi. Par Reiner Sombrowsky.
  - http://www.fichtner.net/delphi/mysql.delphi.phtml Une interface Delphi avec MySQL, avec le code source. Par Matthias Fichtner.
  - http://www.productivity.org/projects/tmysql/ TmySQL, une librairie pour utiliser MySQL avec Delphi.
  - http://sourceforge.net/projects/zeoslib/ La librairie Zeos est un jeu de donnèes et de composants de librairie pour MySql, PostgreSql, Interbase, MS SQL, Oracle, DB/2. Elle inclut aussi des outils de dèveloppement tel que Database Explorer et Database Designer.
  - http://www.mysql.com/Downloads/Contrib/Win32/SBMySQL50Share.exe Composants Shareware Delphi 5 pour MySQL.
- http://www.mysql.com/Downloads/Contrib/mysql-ruby-2.2.0.tar.gz Module MySQL pour Ruby. Par TOMITA Masahiro tommy@tmtm.org Ruby est un langage interprètè orientè objet (http://www.netlab.co.jp/ruby/).
- http://www.mysql.com/Downloads/Contrib/JdmMysqlDriver-0.1.0.tar.gz Un pilote VisualWorks 3.0 Smalltalk pour MySQL. Par joshmiller@earthlink.net.
- http://www.mysql.com/Downloads/Contrib/Db.py Module Python avec cache. Par gandalf@rosmail.com.
- http://www.mysql.com/Downloads/Contrib/MySQLmodule-1.4.tar.gz Interface Python pour MySQL. Par Joseph Skinner joe@earthlight.co.nz. Modifiè par Joerg Senekowitsch senekow@ibm.net.
- http://www.mysql.com/Downloads/Contrib/MySQL-python-0.3.0.tar.gz MySQLdb Python est une interface avec MySQL compatible DB-API v2.0. Les

transactions sont supportèes si le serveur et les tables les supportent. Elle est compatible avec les threads, et dispose d'une interface de compatibilitè avec les anciens codes qui sont èrcit avec l'interface MySQLmodule, aujourd'hui abandonnèe.

- `http://www.mysql.com/Downloads/Contrib/mysql_mex_12.tar.gz` Un programme d'interface pour Matlab de MathWorks. L'interface est faîte par Kimmo Uutela et John Fisher (pas par Mathworks). Vèrifiez `http://boojum.hut.fi/~kuutela/mysqlmex.html` pour plus d'informations.
- `http://www.mysql.com/Downloads/Contrib/mysqltcl-1.53.tar.gz` Interface Tcl pour MySQL. Basèe sur '`msqltcl-1.50.tar.gz`'. Pour la version 2.0 et plus d'informations, voyez `http://www.xdobry.de/mysqltcl/`.
- `http://www.mysql.com/Downloads/Contrib/MyC-0.1.tar.gz` Une API proche du Visual Basic, par Ed Carp.
- `http://www.mysql.com/Downloads/Contrib/Vdb-dflts-2.1.tar.gz` Ceci est la nouvelle version d'un jeu de librairies utilitaires, qui fournissent une interface gènèrique aux moteurs de bases de donnèes, pour que votre application soit entiérement trois tiers. L'avantage est que vous pouvez facilement passer d'un serveur á l'autre en modifiant simplement un fichier, sans toucher á votre application. Par `damian@cablenet.net`.
- `http://www.mysql.com/Downloads/Contrib/DbFramework-1.10.tar.gz` DbFramework est une collection de classes pour interagir avec les bases MySQL. Les classes sont inspirèes par le CDIF Data Model Subject Area. Par Paul Sharpe `paul@miraclefish.com`.
- `http://www.mysql.com/Downloads/Contrib/pike-mysql-1.4.tar.gz` Module MySQL pour pike. A utiliser avec le serveur web Roxen.
- `http://www.mysql.com/Downloads/Contrib/squile.tar.gz` Module pour `guile` qui permet á `guile` de travailler avec les serveurs SQL. Par Hal Roberts.
- `http://www.mysql.com/Downloads/Contrib/stk-mysql.tar.gz` Interface pour Stk. Stk est les widgets Tk qui utilise Scheme au lieu de Tcl. Par Terry Jones.
- `http://www.mysql.com/Downloads/Contrib/eiffel-wrapper-1.0.tar.gz` Module Eiffel par Michael Ravits.
- `http://www.mysql.com/Downloads/Contrib/SQLmy0.06.tgz` FlagShip Replaceable Database Driver (RDD) pour MySQL. Par Alejandro Fernandez Herrero. Le site de Flagship RDD est `http://www.fship.com/rdds.html`.
- `http://www.mysql.com/Downloads/Contrib/mydsn-1.0.zip` Sources et exècutables pour `mydsn.dll`. mydsn doit être utilisèe pour compiler et supprimer le registre DSN pour le pilote MyODBC, pour les applications Coldfusion. Par Miguel Angel Solórzano.
- `http://www.mysql.com/Downloads/Contrib/MySQL-ADA95_API.zip` Une interface ADA95 avec l'API MySQL . Par Francois Fabien.
- `http://www.mysql.com/Downloads/Contrib/MyTool-DLL_for_VB_and_MySQL.zip` Une DLL avec l'API C MySQL pour Visual Basic. Par Ken Menzel `kenm@icarz.com`.
- `http://www.mysql.com/Downloads/Contrib/MYSQLX.EXE` MySQL ActiveX Object pour accèder directement au serveur MySQL depuis un serveur IIS/ASP, VB, VC++ en èvitant les mèthodes ODBC plus lente. Complétement adaptable, multi-threadè,

avec support complet des types de données MySQL (version 2001.1.1). Par SciBit `http://www.scibit.com/`.

- `http://www.fastflow.it/mylua/` MyLUA home page; comment utiliser le langagge LUA pour ècrire des `PROCEDURE` MySQL qui peuvent être exècutè.
  - `http://www.mysql.com/Downloads/Contrib/lua-4.0.tar.gz` LUA 4.0
  - `http://www.mysql.com/Downloads/Contrib/mylua-3.23.32.1.tar.gz` Patch pour MySQL 3.23.32 et LUA 4.0. Par Cristian Giussani.
- `http://www.mysql.com/Downloads/Contrib/patched_myodbc.zip` Patch (pour le support Omniform 4.0) pour le pilote MyODBC. Par Thomas Thaele `tthaele@papenmeier.de`

B.2 Convertisseurs

- `http://www.mysql.com/Downloads/Contrib/mssql2mysql.txt` Convertisseur depuis MS-SQL vers MySQL. Par Michael Kofler. La page de mssql2mysql est `http://www.kofler.cc/mysql/mssql2mysql.html`.
- `http://www.mysql.com/Downloads/Contrib/dbf2mysql-1.14.tar.gz` Convertit des fichiers '`.dbf`' en tables MySQL et vice-versa. Par Maarten Boekhold (`boekhold@cindy.et.tudelft.nl`), William Volkman et Michael Widenius. Ce convertisseur inclut le support rudimentaire des champs MEMO.
- `http://www.mysql.com/Downloads/Contrib/dbf2mysql-1.13.tgz` Convertisseur entre les formats '`.dbf`' et MySQL. Par Maarten Boekhold, `boekhold@cindy.et.tudelft.nl` et Michael Widenius. Ce convertisseur ne peut pas gèrer les champs MEMO.
- `http://www.mysql.com/Downloads/Contrib/dbf2mysql.zip` Convertit les fichiers FoxPro '`.dbf`' en tables MySQL sous Windows. Par Alexander Eltsyn, `ae@nica.ru` ou `ae@usa.net`.
- `http://www.mysql.com/Downloads/Contrib/dbf2sql.zip` Programme court et simple qui vous aide á transporter les donnèes depuis les tables foxpro vers les tables MySQL. Par Danko Josic.
- `http://www.mysql.com/Downloads/Contrib/dump2h-1.20.gz` Convertit le rèsultat de `mysqldump` en fichier d'entête C. Par Harry Brueckner, `brueckner@mail.respublica.de`.
- `http://www.mysql.com/Downloads/Contrib/exportsql.txt` Un script similaire á `access_to_mysql.txt`, hormis le fait que celui-ci est complétement configurable, gére mieux les conversions de type (y compris les champs `TIMESTAMP`), fournit des alertes et des suggestions lors de la conversion, protége **tous** les caractéres spèciaux dans le texte les donnèes binaires, etc. Il va aussi convertir les tables `mSQL` v1 et v2, et il est gratuit pour tous. Voyez `http://www.cynergi.net/exportsql/` pour la derniére version. Par Pedro Freire, `support@cynergi.net`. Note : ne fonctionne pas avec Access2!
- `http://www.mysql.com/Downloads/Contrib/access_to_mysql.txt` Copiez cette fonction dans un module Access qui a accés aux tables que vous voulez exporter. Voyez aussi `exportsql`. Par Brian Andrews. Note: ne fonctionne pas avec Access2!

- `http://www.mysql.com/Downloads/Contrib/importsql.txt` Un script qui fait le contraire de `exportsql.txt`. C'est á dire qu'il importe les donnèes depuis MySQL dans la base Access via ODBC. C'est trés pratique lorsqu'il est compbinè avec exportsql, car il permet d'utiliser Access pour la conception de la base, et la synchronise avec un serveur MySQL. Gratuit. VOyez `http://www.netdive.com/freebies/importsql/` pour les mises á jour. Crèè par Laurent Bossavit of NetDIVE. **Note**: ne fonctionne pas avec Access2!
- `http://www.mysql.com/Downloads/Contrib/mdb2sql.bas` Convertisseur depuis Access97 vers MySQL par Moshe Gurvich.
- `http://www.mysql.com/Downloads/Contrib/msql2mysqlWrapper-1.0.tgz` Une interface C de `mSQL` vers MySQL. Par `alfred@sb.net`
- `http://www.mysql.com/Downloads/Contrib/sqlconv.pl` Un script simple qui peut être utilisè pouar copier des champs d'une table MySQL vers une autre, en masse. En bref, vous pouvez exècuter `mysqldump` et le rediriger vers le script `sqlconv.pl` via un pipe. Le script va analyser le rèsultat de `mysqldump` et va rèarranger les champs pour qu'ils puissent être rèinsèrè dans la nouvelle table. Un exemple d'utilisation est le cas ou vous voulez crèer une nouvelle table á partir d'un site sur lequel vous travaillez, mais que cette table est juste un petit peut diffèrente (au niveau de l'ordre des champs, etc.). Par Steve Shreeve.
- `http://www.mysql.com/Downloads/Contrib/oracledump` Programme Perl pour convertir des bases Oracle vers MySQL. Il a le même format que `mysqldump`. Par Johan Andersson.
- `http://www.mysql.com/Downloads/Contrib/excel2mysql` Programme Perl pour importer des feuilles Excel dans MySQL. Par Stephen Hurd `shurd@sk.sympatico.ca`
- `http://www.mysql.com/Downloads/Contrib/T2S_100.ZIP`. Programme Windows qui convertit des fichiers texte en table MySQL. Par Asaf Azulay.

B.3 Utilitaire

- `http://worldcommunity.com/opensource/utilities/mysql_backup.html` MySQL Backup est un script de sauvegarde pour MySQL. Par Peter F. Brown.
- `http://www.mysql.com/Downloads/Contrib/mysql_watchdog.pl` Surveille le dèmon MySQL contre les blocages. Par Yermo Lamers, `yml@yml.com`.
- `http://www.mysql.com/Downloads/Contrib/mysql_structure_dumper.tar.gz`
- `http://www.mysql.com/Downloads/Contrib/mysql_structure_dumper.tgz` Affiche la structure de toutes les tables dans une base. Par Thomas Wana.
- `http://www.mysql.com/Downloads/Contrib/mysqlsync`. Un script Perl pour synchroniser des tables MySQL distantes, avec une copie centrale.Par Mark Jeftovic. `markjr@easydns.com`.
- `http://www.mysql.com/Downloads/Contrib/MySQLTutor-0.2.tar.gz`. MySQLTutor. Un tutoriel MySQL pour les dèbutants.
- `http://www.mysql.com/Downloads/Contrib/MySQLDB.zip`
- `http://www.mysql.com/Downloads/Contrib/MySQLDB-readme.html`. Une librairie COM pour MySQL par Alok Singh.

- `http://www.mysql.com/Downloads/Contrib/mysql_replicate.pl` Un programme Perl qui gére les rèplications. Par `elble@icculus.nsg.nwu.edu`
- `http://www.mysql.com/Downloads/Contrib/DBIx-TextIndex-0.02.tar.gz` Un script Perl qui utilise l'indexation inversse pour gèrer les recherches dans les grands textes. Par Daniel Koch.
- `http://www.mysql.com/Downloads/Contrib/dbcheck` Un script Perl qui installe des copies de sauvegarde, et exècute `isamchk` dessus. Par Elizabeth.
- `http://www.mysql.com/Downloads/Contrib/mybackup`.
- `http://www.mswanson.com/mybackup` (mybackup home page) Surcouche de `mysqldump` qui sauvegarde toutes les bases. Par Marc Swanson.
- `http://www.mysql.com/Downloads/Contrib/mdu.pl.gz` Affiche la place prise par une base MySQL.

Annexe C Crèdits

Cet annexe liste les dèveloppeurs, contributeurs, et supporters qui ont fait de MySQL est ce qu'il est aujourd'hui.

C.1 Dèveloppeurs chez MySQL AB

Voici les dèveloppeurs qui sont employès par `MySQL AB` pour travailler sur la base de donnèes `MySQL`, grosso modo dans leur ordre d'embauche. Pour chaque dèveloppeur, vous trouverez une petite liste de leur tâches et de leur responsabilitès, ainsi que leur rèalisations. Tous les dèveloppeurs sont impliquès dans le support.

Michael (Monty) Widenius

- Chef d'èquipe et auteur principal du serveur MySQL (`mysqld`).
- Nouvelles fonctions dans la librairie de chaînes.
- La majoritè de la librairie `mysys`.
- Les librairies `ISAM` et `MyISAM` (les fichiers d'index B-tree avec compression d'index et diffèrentes formats d'enregistrement).
- La librairie `HEAP`. Un systéme de gestion des tables en mèmoire avec un hashage dynamique efficace. Utilisè depuis 1980 et publiè en 1984.
- Le programme `replace` (jetez y un oeil c'est carrèment **COOL**!).
- `MyODBC`, le pilote ODBC de Windows95.
- La correction de bogues avec les MIT-pthreads pour qu'ils fonctionnent avec MySQL. Et aussi l'Unireg, une application á base de curses avec de nombreuses utilisations.
- Le port d'outils `mSQL` comme `msqlperl`, `DBD/DBI` et `DB2mysql`.
- La majoritè de `crash-me` et les fondations des tests de performances de MySQL.

David Axmark

- Auteur original du **manuel de rèfèrence**, incluent les amèliorations de `texi2html`.
- La mise á jour automatique du site web depuis le manuel.
- Le support initial de Autoconf, Automake et Libtool.
- Les licences.
- Des parties de tous les fichiers textes (de nos jours, uniquement le '`README`' est encore prèsent. le reste est dans le manuel).
- Nombreux tests des nouvelles fonctionnalitès.
- Notre expert lègal des logiciels libres.
- Le responsable des listes de diffusion (qui n'a jamais le temps de le faire correctement...).
- Notre code initial pour le port (bientôt plus de 10 ans). De nos jours, seules des parties de `mysys` restent.

- La personne que Monty appelle au milieu de la nuit lorsqu'il a rèussi a faire fonctionner la nouvelle fonctionnalitè!
- Le chef "Open Source" (Relations avec la communautè MySQL).

Jani Tolonen

- `mysqlimport`
- Un grand nombre d'extension pour le client en ligne de commande.
- `PROCEDURE ANALYSE()`

Sinisa Milivojevic

- Compression du protocole client/serveur avec `zlib`.
- Hashing parfait pour la phase d'analyse lexicale.
- Insertions multi-lignes
- Option `mysqldump` -e
- `LOAD DATA LOCAL INFILE`
- Option `SQL_CALC_FOUND_ROWS SELECT`
- Option `--max-user-connections=...`
- `net_read` et `net_write_timeout`
- `GRANT/REVOKE` et `SHOW GRANTS FOR`
- Nouveau protocole client/serveur pour la version 4.0
- `UNION` en version 4.0
- Traitements multi-table de `DELETE/UPDATE`
- Tables dèrivèes en version 4.1
- Gestion des ressources utilisateurs.
- Dèveloppeur initial de l'API `MySQL++` C++ API et du client `MySQLGUI`.

Tonu Samuel (ancien dèveloppeur)

- Interface VIO (la base pour le protocole client/serveur chiffrè).
- Systéme de fichier MySQL (une mèthode pour utiliser la base MySQL comme un systéme de fichiers).
- L'expression `CASE`.
- Les fonctions `MD5()` et `COALESCE()`.
- Le support `RAID` des tables `MyISAM`.

Sasha Pachev

- Implèmentation initiale de la rèplication (jusqu'en version 4.0).
- `SHOW CREATE TABLE`.
- `mysql-bench`

Matt Wagner

- Suite de tests MySQL.
- Webmestre (jusqu'en 2002).
- Coordinateur du dèveloppement.

Miguel Solorzano

- Dèveloppement Win32 et publications.
- Code du serveur sur Windows NT.
- WinMySQLAdmin

Timothy Smith (ancien dèveloppeur)

- Support des jeux de caractéres dynamiques.
- le script configure, les RPMs et d'autres parties du systéme de compilation.
- Dèveloppeur initial de `libmysqld`, le serveur intègrè.

Sergei Golubchik

- Recherche en texte plein.
- Librairie de clès pour `MERGE`.

Jeremy Cole

- Relecture et èdition de ce manuel.
- `ALTER TABLE ... ORDER BY ...`.
- `UPDATE ... ORDER BY ...`.
- `DELETE ... ORDER BY ...`.

Indrek Siitan

- Design et programmation de notre interface web.
- Auteur de notre lettre d'actualitè.

Jorge del Conde

- `MySQLCC` (`MySQL Control Center`)
- Dèveloppement Win32
- Implèmentation initiale des portails du site web.

Venu Anuganti

- MyODBC 3.51
- Nouveau protocole client/serveur pour la version 4.1

Arjen Lentz

- Responsable du manuel de rèfèrence MySQL.
- Prèparation de la version imprimèe chez O'Reilly.

Alexander (Bar) Barkov, Alexey (Holyfoot) Botchkov et Ramil Kallimullin

- Donnèes spatiales (GIS) et R-Trees en version 4.1
- Unicode et jeux de caractéres pour la version 4.1

Alexander (Sanja) Belkin

- Cache de requêtes en version 4.0
- Implèmentation des sous-requêtes en version 4.1.

Aleksey (Walrus) Kishkin et Alexey (Ranger) Stroganov

- Design des tests de performance et analyse.
- Maintenance de la suite de test MySQL.

Zak Greant
- Porte parole Open Source, relation avec la communautè.

Carsten Pedersen
- Le programme de certification MySQL.

Lenz Grimmer
- Ingènierie de mise en production (compilation et publication).

Peter Zaitsev
- Fonction `SHA1()`, `AES_ENCRYPT()` et `AES_DECRYPT()`.
- Dèbogage et nettoyage de diverses fonctionnalitès.

Alexander (Salle) Keremidarski
- Documentation du code et des algorithmes MySQL.
- Dèbogage.

Per-Erik Martin
- Chef de projet pour les procèdures stockèes et les triggers.

Jim Winstead
- Chef dèveloppeur web.

Mark Matthews
- Pilote JDBC.

C.2 Contributeurs á MySQL

Bien que `MySQL AB` posséde tous les droits du `serveur MySQL` et du `manuel MySQL`, nous voulons montrer notre reconnaissance á ceux qui ont apportès leur contribution d'une maniére ou d'une autre á la `distribution MySQL`. Les contributeurs sont listès ici, dans un ordre alèatoire :

Paul DuBois
: Aide constante pour rendre le manuel correct et comprèhensible. Ce qui inclus la rèècriture de l'anglais de Monty et David's en un anglais que tout le monde comprend.

Gianmassimo Vigazzola `qwerg@mbox.vol.it` ou `qwerg@tin.it`
: Le portage initial sur Win32/NT.

Kim Aldale
: Aide á la rèècriture de l'anglais de Monty et David.

Per Eric Olsson
: Pour des critiques plus ou moins constructives, et des tests sur les formats de lignes dynamiques.

Irena Pancirov `irena@mail.yacc.it`
: Portage sur Win32 avec le compilateur Borland. `mysqlshutdown.exe` et `mysqlwatch.exe`

David J. Hughes
: Pour ses efforts de constitution d'une base de donnèes SQL partagèe. Chez TcX, le prèdècesseur de MySQL AB, nous avons commencè avec `mSQL`, mais nous nous sommes aperçus que cela ne satisfaisait pas nos besoins, et nous avons ècrit une interface SQL avec notre application Unireg. Les clients `mysqladmin` et `mysql` ont ètè largement influencès par leur èquivalent `mSQL`. Nous avons mis une grande partie de nos efforts á faire que la syntaxe de MySQL soit un sur ensemble de celle de `mSQL`. De nombreuses idèes d'API ont ètè empruntèes á `mSQL` pour rendre plus facile le portage des programmes depuis `mSQL` vers MySQL. Le logiciel MySQL ne contient aucun code extrait de `mSQL`. Les deux fichiers ('`client/insert_test.c`' et '`client/select_test.c`') qui font partie de la distribution sont basès sur les fichiers correspondants, et sans droits de la distribution `mSQL`, mais sont modifiès, pour servir d'exemple aux modifications nècessaires á la conversion de code de `mSQL` vers MySQL. (`mSQL` est la propriètè de David J. Hughes).

Fred Fish
: Pour son excellente librairie de dèbogage et de trace en langage C. Monty a fait un grand nombre de petites amèliorations de cette librairie (en vitesse et en fonctionnalitès).

Richard A. O'Keefe
: Pour sa librairie de chaînes de caractéres du domaine public.

Henry Spencer
: Pour sa librairie d'expression règuliéres, utilisèes dans les clauses `WHERE column REGEXP regexp`.

Free Software Foundation
: De qui nous tenons l'excellent compilateur (`gcc`), la librairie `libc` (dont nous avons empruntè le fichier '`strto.c`' pour faire fonctionner du code sous Linux), et pour la librairie `readline` (pour le client `mysql`).

Free Software Foundation & The XEmacs development team
: Pour un environnement et un èditeur vraiment excellent, utilisè pratiquement par tout le monde á MySQL AB/TcX/detron.

Patrick Lynch
: Pour nous avoir aidè á acquèrir `http://www.mysql.com/`.

Fred Lindberg
: Pour avoir configurè qmail pour qu'il gére les listes de diffusions MySQLm et pour l'aide incroyable que nous avons reçu pour gèrer les listes de diffusions de MySQL.

Igor Romanenko `igor@frog.kiev.ua`
: `mysqldump` (prècèdemment appelè `msqldump`, mais portè et amèliorè par Monty).

Yuri Dario
: Pour avoir suivi et amèliorè la version MySQL pour OS/2.

Tim Bunce, Alligator Descartes
: Pour l'interface `DBD` (Perl).

Tim Bunce
: Auteur de `mysqlhotcopy`.

Andreas Koenig `a.koenig@mind.de`
: Pour l'interface Perl avec le serveur MySQL.

Eugene Chan `eugene@acenet.com.sg`
: Pour avoir portè MySQL avec PHP.

Michael J. Miller Jr. `mke@terrapin.turbolift.com`
: Pour le premier manuel MySQL. Et un grand nombre de corrections d'erreurs et d'orthographe dans la FAQ (qui est devenu le manuel MySQL depuis bien longtemps).

Yan Cailin
: Premier traducteur du manuel de rèfèrence MySQL en chinois simplifiè, au dèbut de l'an 2000, sur lequel les versions Big5 et HK (`http://mysql.hitstar.com/`) sont basèes. Personal home page at linuxdb.yeah.net (`http://linuxdb.yeah.net`).

Giovanni Maruzzelli `maruzz@matrice.it`
: Pour le portage de iODBC (Unix ODBC).

Chris Provenzano
: pthreads portables au niveau utilisateur. D'aprés les droits : ce produit inclut un logiciel dèveloppè par Chris Provenzano, l'University of California, Berkeley et des contributeurs. Nous utilisons actuellement la version 1_60_beta6, modifièe par Monty (voir '`mit-pthreads/Changes-mysql`').

Xavier Leroy `Xavier.Leroy@inria.fr`
: L'auteur de LinuxThreads (utilisè par le serveur MySQL sous Linux).

Zarko Mocnik `zarko.mocnik@dem.si`
: Tri pour le slovéne et le module '`cset.tar.gz`' qui rend plus simple l'ajout d'autres jeux de caractéres.

"TAMITO" `tommy@valley.ne.jp`
: Les macros de jeu de caractéres `_MB`m et les jeux de caractéres ujis et sjis.

Joshua Chamas `joshua@chamas.com`
: La base des insertions concurrentes, la syntaxe de date amèliorèe, le dèbogage sous NT et les rèponses sur la liste de diffusion.

Yves Carlier `Yves.Carlier@rug.ac.be`
: `mysqlaccess`, un programme qui affiche les droits des utilisateurs.

Rhys Jones `rhys@wales.com` (And GWE Technologies Limited)
: Pour JDBC, un module pour extraire des donnèes de bases MySQL avec un client Java.

Dr Xiaokun Kelvin ZHU `X.Zhu@brad.ac.uk`
: Dèveloppement du pilote JDBC et d'autres outils MySQL liès á Java.

James Cooper `pixel@organic.com`
: Pour la configuration d'une archive indexèe des listes de diffusion sur son site.

Rick Mehalick `Rick_Mehalick@i-o.com`
: Pour `xmysql`, un client grahique X pour MySQL.

Doug Sisk `sisk@wix.com`
: Pour les packages RPM de MySQL pour RedHat Linux.

Diemand Alexander V. `axeld@vial.ethz.ch`
: Pour les packages RPM de MySQL pour RedHat Linux-Alpha.

Antoni Pamies Olive `toni@readysoft.es`
: Pour les packages RPM de MySQL pour Intel et SPARC.

Jay Bloodworth `jay@pathways.sde.state.sc.us`
: Pour les packages RPM de MySQL pour MySQL Version 3.21.

Jochen Wiedmann `wiedmann@neckar-alb.de`
: Pour l'entretien du module Perl `DBD::mysql`.

Therrien Gilbert `gilbert@ican.net`, Jean-Marc Pouyot `jmp@scalaire.fr`
: Pour les messages d'erreurs en français.

Petr Snajdr, `snajdr@pvt.net`
: Pour les messages d'erreurs en tchéque.

Jaroslaw Lewandowski `jotel@itnet.com.pl`
: Pour les messages d'erreurs en polonais.

Miguel Angel Fernandez Roiz
: Pour les messages d'erreurs en espagnol.

Roy-Magne Mo `rmo@www.hivolda.no`
: Pour les messages d'erreurs en norvègien, et les tests de la version 3.21.#.

Timur I. Bakeyev `root@timur.tatarstan.ru`
: Pour les messages d'erreurs en russe.

`brenno@dewinter.com` & Filippo Grassilli `phil@hyppo.com`
: Pour les messages d'erreurs en italien.

Dirk Munzinger `dirk@trinity.saar.de`
: Pour les messages d'erreurs en allemand.

Billik Stefan `billik@sun.uniag.sk`
: Pour les messages d'erreurs en slovéne.

Stefan Saroiu `tzoompy@cs.washington.edu`
: Pour les messages d'erreurs en roumain.

Peter Feher
: Pour les messages d'erreurs en hongrois.

Roberto M. Serqueira
: Pour les messages d'erreurs en portugais.

Carsten H. Pedersen
: Pour les messages d'erreurs en danois.

Arjen G. Lentz
: Pour les messages d'erreurs en hollandais, en complèment d'une traduction initiale partielle, et d'un travail sur la cohèrence et l'orthographe.

David Sacerdote `davids@secnet.com`
: Idèes pour la vèrification sècuritaire des noms d'hôtes.

Wei-Jou Chen `jou@nematic.ieo.nctu.edu.tw`
: Le support des caractéres chinois Chinese(BIG5).

Wei He `hewei@mail.ied.ac.cn`
: Un grand nombre de fonctionnalitès pour le jeu de caractéres Chinese(GBK).

Zeev Suraski `bourbon@netvision.net.il`
: Format d'heure `FROM_UNIXTIME()`, fonctions `ENCRYPT()` et conseiller `bison`. Membre actif des listes de diffusion.

Luuk de Boer `luuk@wxs.nl`
: Portage et amèlioration de la suite de tests avec `DBI/DBD`. A ètè d'une grande aide avec le test `crash-me` et l'exècution des tests. Certaines amèliorations de la fonction de date. Le script `mysql_setpermissions`.

Jay Flaherty `fty@mediapulse.com`
: Grand parties de la section Perl `DBI/DBD` dans le manuel.

Paul Southworth `pauls@etext.org`, Ray Loyzaga `yar@cs.su.oz.au`
: Relecture du manuel.

Alexis Mikhailov `root@medinf.chuvashia.su`
: Fonctions utilisateurs (UDF); `CREATE FUNCTION` et `DROP FUNCTION`.

Andreas F. Bobak `bobak@relog.ch`
: L'extension `AGGREGATE` des fonctions UDF.

Ross Wakelin `R.Wakelin@march.co.uk`
: Aide avec InstallShield pour MySQL-Win32.

Jethro Wright III `jetman@li.net`
: La librairie '`libmysql.dll`'.

James Pereria `jpereira@iafrica.com`
: Mysqlmanager, un outil d'administration Win32 graphique pour MySQL.

Curt Sampson `cjs@portal.ca`
: Portage des MIT-pthreads vers NetBSD/Alpha et NetBSD 1.3/i386.

Antony T. Curtis `antony.curtis@olcs.net`
: Portage de la base MySQL sur OS/2.

Martin Ramsch `m.ramsch@computer.org`
: Exemples dans les tutoriels du manuel MySQL.

Steve Harvey
: Pour la sècurisation de `mysqlaccess`.

Konark IA-64 Centre of Persistent Systems Private Limited
: `http://www.pspl.co.in/konark/`. Aide avec le port Win64 du serveur MySQL.

Albert Chin-A-Young.
: Modifications de la configuration pour Tru64, support des grands fichiers, et amèlioration des gestionnaires TCP.

John Birrell
: Emulation de `pthread_mutex()` pour OS/2.

Benjamin Pflugmann
: Tables `MERGE` amèliorèe pour la gestion des `INSERTS`. Membre actif des listes de diffusion.

Guilhem Bichot
: Correction des exposants du type `DECIMAL`. Auteur de `mysql_tableinfo`.

D'auters contributeurs, dèbusqueurs de bogues et testeurs : James H. Thompson, Maurizio Menghini, Wojciech Tryc, Luca Berra, Zarko Mocnik, Wim Bonis, Elmar Haneke, `jehamby@lightside`, `psmith@BayNetworks.com`, `duane@connect.com.au`, Ted Deppner `ted@psyber.com`, Mike Simons, Jaakko Hyvatti.

Et bien d'autres rapports de bogues et de corrections, fournis par les amis des listes de diffusions :

Notre reconnaissance va á ceux qui nous aident á rèpondre sur les listes de diffusions `mysql@lists.mysql.com` :

Daniel Koch `dkoch@amcity.com`
: Configuration Irix.

Luuk de Boer `luuk@wxs.nl`
: Questions de tests de performances.

Tim Sailer `tps@users.buoy.com`
: Questions sur `DBD-mysql`.

Boyd Lynn Gerber `gerberb@zenez.com`
: Questions sur SCO.

Richard Mehalick `RM186061@shellus.com`
: Questions sur `xmysql` et les questions d'installation simples.

Zeev Suraski `bourbon@netvision.net.il`
: Questions sur le module apache module et sa configuration (log & auth), Questions sur PHP, la syntaxe SQL, et d'autres questions en gènèral.

Francesc Guasch `frankie@citel.upc.es`
: Questions gènèrales.

Jonathan J Smith `jsmith@wtp.net`
: Questions sur les spècificitès des Linux, la syntaxe SQL, et d'autres aspects qui requiert un peu de travail.

David Sklar `sklar@student.net`
: Utilisation de MySQL avec PHP et Perl.

Alistair MacDonald `A.MacDonald@uel.ac.uk`
: Pas vraiment prècisè, mais gènèralise, et sais gérer les questions Linux et un peu HP-UX. Pousse l'utilisateur á exploiter `mysqlbug`.

John Lyon `jlyon@imag.net`
: Questions sur l'installation de MySQL sur Linux, avec les paquets '`.rpm`' ou via les sources.

Lorvid Ltd. `lorvid@WOLFENET.com`
: Questions simples sur les facturations, les licences, le support et les droits.

Patrick Sherrill `patrick@coconet.com`
: Questions sur les interfaces ODBC et VisualC++.

Randy Harmon `rjharmon@uptimecomputers.com`
: Questions sur `DBD`, Linux, et la syntaxe SQL.

C.3 Supporters de MySQL

Tandis que `MySQL AB` posséde tous les copyrights du `serveur MySQL` et du `manuel MySQL`, nous voulons montrer notre reconnaissances aux compagnies suivantes, qui nous ont aidè á financer le developement du `serveur MySQL`, soit en nous payant pour developper de nouvelles fonctionnalitèes, soit en nous fournissant en matèriel pour le developpement du `serveur MySQL`.

VA Linux / Andover.net
: Financement de la replication.

NuSphere
: Edition du manuel MySQL.

Stork Design studio
: Le site web MySQL utilisè entre 1998 et 2000.

Intel
: Contribution au dèveloppement sur les plate-formes Windows et Linux.

Compaq
: Contribution au dèveloppement sur Linux/Alpha.

SWSoft
: Dèveloppement de la version embarquèe `mysqld`.

FutureQuest
: `--skip-show-database`

Annexe D Historique des changements MySQL

Cet appendice liste les changements de version á version dans le code source de MySQL.

Nous travaillons maintenant activement sur MySQL 4.x et ne fournirons que les correctifs pour les bogues critiques de MySQL 3.23. Nous mettons á jour cette section lorsque nous ajoutons de nouvelles fonctionnalitès pour que tout le monde puisse suivre le cours du dèveloppement.

Notre section TODO contient ce que nous planifions pour les versions 4.x. Voir Section 1.8 [TODO], page 46.

Notez que nous essayons de mettre á jour le manuel en même temps que nous apportons des changements á MySQL. Si vous trouvez une version mentionnèe ici que vous ne pouvez retrouver dans la page des tèlèchargements MySQL (`http://www.mysql.com/downloads/`), cela signifie que la version n'a pas encore ètè publiée !

D.1 Changes in release 4.1.x (Alpha)

Version 4.1 of the MySQL server includes many enhancements and new features:

- Subqueries: `SELECT * from t1 where t1.a=(SELECT t2.b FROM t2)`.
- Derived tables: `SELECT a from t1, (select * from t2) WHERE t1.a=t2.a`
- Character sets to be defined per column, table and database.
- Unicode (UTF8) support.
- `BTREE` index on `HEAP` tables.
- Support for GIS (Geometrical data).
- `SHOW WARNINGS`; Shows warnings for the last command.

For a full list of changes, please refer to the changelog sections for each individual 4.1.x release.

D.1.1 Changes in release 4.1.0

- Fixed `SELECT .. LIMIT 0` to return proper row count for `SQL_CALC_FOUND_ROWS`.
- One can specify many temporary directories to be used in a round-robin fashion with: `--tmpdir=dirname1:dirname2:dirname3`.
- Subqueries: `SELECT * from t1 where t1.a=(SELECT t2.b FROM t2)`.
- Derived tables: `SELECT a from t1, (select * from t2) WHERE t1.a=t2.a`
- Character sets to be defined per column, table and database.
- Unicode (UTF8) support.
- `BTREE` index on `HEAP` tables.
- Faster embedded server.
- One can add a comment per column in `CREATE TABLE`.
- `SHOW FULL COLUMNS FROM table_name` shows column comments.
- `ALTER DATABASE`.

- Support for GIS (Geometrical data).
- `SHOW WARNINGS`; Shows warnings from the last command.
- One can specify a column type for a colum in `CREATE TABLE ... SELECT` by defining the column in the `CREATE` part.

```
CREATE TABLE foo (a tinyint not null) SELECT b+1 AS 'a' FROM bar;
```

D.2 Changes in release 4.0.x (Beta)

Version 4.0 of the MySQL server includes many enhancements and new features:

- The `InnoDB` table type is now included in the standard binaries, adding transactions, row-level locking, and foreign keys. Voir Section 7.5 [InnoDB], page 544.
- A query qache, offering vastly increased performance for many applications. By caching complete result sets, later identical queries can return instantly. Voir Section 6.9 [Query Cache], page 527.
- Improved full-text indexing with boolean mode, truncation, and phrase searching. Voir Section 6.8 [Fulltext Search], page 522.
- Enhanced `MERGE` tables, now supporting `INSERT`s and `AUTO_INCREMENT`. Voir Section 7.2 [MERGE], page 539.
- `UNION` syntax in `SELECT`. Voir Section 6.4.1.2 [UNION], page 487.
- Multi-table `DELETE` statements. Voir Section 6.4.6 [DELETE], page 494.
- `libmysqld`, the embedded server library. Voir Section 8.4.9 [libmysqld], page 662.
- Additional `GRANT` privilege options for even tighter control and security. Voir Section 4.3.1 [GRANT], page 226.
- Management of user resources in the `GRANT` system, particularly useful for ISPs and other hosting providers. Voir Section 4.3.6 [User resources], page 236.
- Dynamic server variables, allowing configuration changes without having to take down the server. Voir Section 5.5.6 [SET OPTION], page 396.
- Improved replication code and features. Voir Section 4.10 [Replication], page 332.
- Numerous new functions and options.
- Changes to existing code for enhanced performance and reliability.

For a full list of changes, please refer to the changelog sections for each individual 4.0.x release.

D.2.1 Changes in release 4.0.6

Functionality added or changed:

Bugs fixed:

- Fixed a bug that caused `OPTIMIZE TABLE` to corrupt the table under some rare circumstances.

D.2.2 Changes in release 4.0.5 (13 Nov 2002)

Functionality added or changed:

- Added `Qcache_lowmem_prunes` status variable (number of queries that were deleted from cache because of low memory).
- Fixed `mysqlcheck` so it can deal with table names containing dashes.
- Changed handling of last argument in `WEEK()` so that one can get week number according to the ISO 8601 specification. (Old code should still work).
- Fixed that `INSERT DELAY` threads doesn't hang on `Waiting for INSERT` when one sends a `SIGHUP` to `mysqld`.
- Change that `AND` works according to SQL ANSI99 when it comes to `NULL` handling. In practice, this only affects queries where you do something like `WHERE ... NOT (NULL AND 0)`.
- `mysqld` will now resolve `basedir` to its full path (with `realpath()`). This enables one to use relative symlinks to the MySQL installation directory. This will however cause `show variables` to report different directories on systems where there is a symbolic link in the path.
- Fixed that MySQL will not use an index scan on an index that has been disabled with `IGNORE INDEX` or `USE INDEX`.
- Added `--use-frm` option to `mysqlcheck`. When used with `REPAIR`, it gets the table structure from the `.frm` file, so the table can be repaired even if the `.MYI` header is corrupted.
- Fixed bug in `MAX()` optimization when used with `JOIN` and `ON` expressions.
- Added support for reading of MySQL 4.1 table definition files.
- `BETWEEN` behaviour changed (voir Section 6.3.1.2 [Comparison Operators], page 437). Now `datetime_col BETWEEN timestamp AND timespamp` should work as expected.
- One can create `TEMPORARY MERGE` tables now.
- `DELETE FROM myisam_table` now shrinks not only the '`.MYD`' file but also the '`.MYI`' file.
- When one uses the `--open-files-limit=#` option to `mysqld_safe` it's now passed on to `mysqld`.
- Changed output from `EXPLAIN` from `'where used'` to `'Using where'` to make it more in line with other output.
- Removed variable `safe_show_database` as it was not used anymore.
- Updated source tree to be built using `automake 1.5` and `libtool 1.4`.
- Fixed an inadvertently changed option (`--ignore-space`) back to the original `--ignore-spaces` in `mysqlclient`. (Both syntaxes will work).
- Don't require `UPDATE` privilege when using `REPLACE`.
- Allow braces in joins in all positions. Formerly, things like `SELECT * FROM (t2 LEFT JOIN t3 USING (a)), t1` worked, but not `SELECT * FROM t1, (t2 LEFT JOIN t3 USING (a))`. Note that braces are simply removed, they do not change the way the join is executed.

- InnoDB now supports also isolation levels `READ UNCOMMITTED` and `READ COMMITTED`. For a detailed InnoDB changelog, see the section InnoDB Change History in this manual.

Bugs fixed:

- Corrected test for `root` user in `mysqld_safe`.
- Fixed error message issued when table handler cannot do `CHECK` or `REPAIR`.
- Fixed rare core dump problem in complicated `GROUP BY` queries that didn't return any result.
- Fixed `mysqlshow` to work properly with wildcarded database names and with database names that contain underscores.
- Added support for `DROP TEMPORARY TABLE ...`, to be used to make replication safer.
- When transactions are enabled, all commands that update temporary tables inside a `BEGIN/COMMIT` are now stored in the binary log on `COMMIT` and not stored if one does `ROLLBACK`. This fixes some problems with non-transactional temporary tables used inside transactions.
- Portability fixes to get MySQL to compile cleanly with Sun Forte 5.0.
- Fixed `MyISAM` crash when using dynamic-row tables with huge numbers of packed fields.
- Fixed query cache behaviour with `BDB` transactions.
- Fixed possible floating point exception in `MATCH` relevance calculations.
- Fixed bug in full-text search `IN BOOLEAN MODE` that made `MATCH` to return incorrect relevance value in some complex joins.
- Fixed a bug that limited `MyISAM` key length to a value slightly less that 500. It is exactly 500 now.
- Fixed that `GROUP BY` on columns that may have a `NULL` value doesn't always use disk based temporary tables.
- The filename argument for the `--des-key-file` argument to `mysqld` is interpreted relative to the data directory if given as a relative pathname.
- Removed a condition that temp table with index on column that can be `NULL` has to be `MyISAM`. This was okay for 3.23, but not needed in 4.*. This resulted in slowdown in many queries since 4.0.2.
- Small code improvement in multi-table updates.
- Fixed a newly introduced bug that caused `ORDER BY ... LIMIT #` to not return all rows.
- Fixed a bug in multi-table deletes when outer join is used on an empty table, which gets first to be deleted.
- Fixed a bug in multi-table updates when a single table is updated.
- Fixed bug that caused `REPAIR TABLE` and `myisamchk` to corrupt `FULLTEXT` indexes.
- Fixed bug with caching the `mysql` grant table database. Now queries in this database are not cached in the query cache.
- Small fix in `mysqld_safe` for some shells.
- Give error if a `MyISAM MERGE` table has more than 2 ^ 32 rows and MySQL was not compiled with with `-DBIG_TABLES`.
- Fixed some `ORDER BY ... DESC` problems with `InnoDB` tables.

D.2.3 Changes in release 4.0.4 (29 Sep 2002)

- Fixed bug where `GRANT/REVOKE` failed if hostname was given in non-matching case.
- Don't give warning in `LOAD DATA INFILE` when setting a `timestamp` to a string value of `'0'`.
- Fixed bug in `myisamchk -R` mode.
- Fixed bug that caused `mysqld` to crash on `REVOKE`.
- Fixed bug in `ORDER BY` when there is a constant in the `SELECT` statement.
- One didn't get an error message if `mysqld` couldn't open the privilege tables.
- `SET PASSWORD FOR ...` closed the connection in case of errors (bug from 4.0.3).
- Increased max possible `max_allowed_packet` in `mysqld` to 1GB.
- Fixed bug when doing a multi-line `INSERT` on a table with an `AUTO_INCREMENT` key which was not in the first part of the key.
- Changed `LOAD DATA INFILE` to not recreate index if the table had rows from before.
- Fixed overrun bug when calling `AES_DECRYPT()` with incorrect arguments.
- `--skip-ssl` can now be used to disable SSL in the MySQL clients, even if one is using other SSL options in an option file or previously on the command line.
- Fixed bug in `MATCH ... AGAINST( ... IN BOOLEAN MODE)` used with `ORDER BY`.
- Added `LOCK TABLES` and `CREATE TEMPORARY TABLES` privilege on the database level. One must run the `mysql_fix_privilege_tables` script on old installations to activate these.
- In `SHOW TABLE ... STATUS`, compressed tables sometimes showed up as `dynamic`.
- `SELECT @@[global|session].var_name` didn't report `global | session` in the result column name.
- Fixed problem in replication that `FLUSH LOGS` in a circular replication setup created an infinite number of binary log files. Now a `rotate-binary-log` command in the binary log will not cause slaves to rotate logs.
- Removed `STOP EVENT` from binary log when doing `FLUSH LOGS`.
- Disable the use of `SHOW NEW MASTER FOR SLAVE` as this needs to be completely changed in 4.1.
- Fixed a bug with constant expression (e.g. field of a one-row table, or field from a table, referenced by a `UNIQUE` key) appeared in `ORDER BY` part of `SELECT DISTINCT`.
- `--log-binary=a.b.c` now properly strips off `.b.c`.
- `FLUSH LOGS` removed numerical extension for all future update logs.
- `GRANT ... REQUIRE` didn't store the SSL information in the `mysql.user` table if SSL was not enabled in the server.
- `GRANT ... REQUIRE NONE` can now be used to remove SSL information.
- `AND` is now optional between `REQUIRE` options.
- `REQUIRE` option was not properly saved, which could cause strange output in `SHOW GRANTS`.
- Fixed that `mysqld --help` reports correct values for `--datadir` and `--bind-address`.

- Fixed that one can drop UDFs that didn't exist when `mysqld` was started.
- Fixed core dump problem with `SHOW VARIABLES` on some 64 bit systems (like Solaris sparc).
- Fixed a bug in my_getopt; `--set-variable` syntax didn't work for those options that didn't have a valid variable in my_option struct. This affected at least `default-table-type` option.
- Fixed a bug from 4.0.2 that caused `REPAIR TABLE` and `myisamchk --recover` to fail on tables with duplicates in a unique key.
- Fixed a bug from 4.0.3 in calculating the default field type for some functions. This affected queries of type `CREATE TABLE table_name SELECT expression(),...`
- Fixed bug in queries of type `SELECT * FROM table-list GROUP BY ...` and `SELECT DISTINCT * FROM ...`.
- Fixed bug with the `--slow-log` when logging an administrator command (like `FLUSH TABLES`).
- Fixed a bug that `OPTIMIZE` of locked and modified table, reported table corruption.
- Fixed a bug in my_getopt in handling of special prefixes (`--skip-`, `--enable-`). `--skip-external-locking` didn't work and the bug may have affected other similar options.
- Fixed bug in checking for output file name of the `tee` option.
- Added some more optimisation to use index for `SELECT ... FROM many_tables .. ORDER BY key limit #`
- Fixed problem in `SHOW OPEN TABLES` when a user didn't have access permissions to one of the opened tables.

D.2.4 Changes in release 4.0.3 (26 Aug 2002: Beta)

- Fixed problem with `configure ... --localstatedir=...`.
- Cleaned up `mysql.server` script.
- Fixed a bug in `mysqladmin shutdown` when pid file was modified while `mysqladmin` was still waiting for the previous one to disappear. This could happen during a very quick restart and caused `mysqladmin` to hang until `shutdown_timeout` seconds had passed.
- Don't increment warnings when setting `AUTO_INCREMENT` columns to `NULL` in `LOAD DATA INFILE`.
- Fixed all boolean type variables/options to work with the old syntax, e.g. all of these work: –lower-case-table-names, –lower-case-table-names=1, -O lower-case-table-names=1, –set-variable=lower-case-table-names=1
- Fixed shutdown problem (SIGTERM signal handling) on Solaris. (Bug from 4.0.2).
- `SHOW MASTER STATUS` now returns an empty set if binary log is not enabled.
- `SHOW SLAVE STATUS` now returns an empty set if slave is not initialised.
- Don't update MyISAM index file on update if not strictly necessary.
- Fixed bug in `SELECT DISTINCT ... FROM many_tables ORDER BY not-used-column`.

- Fixed a bug with `BIGINT`s and quoted strings.
- Added `QUOTE()` function that performs SQL quoting to produce values that can be used as data values in queries.
- Changed variable `DELAY_KEY_WRITE` to an enum to allow one set `DELAY_KEY_WRITE` for all tables without taking down the server.
- Changed behaviour of `IF(condition,column,NULL)` so that it returns the value of the column type.
- Made `safe_mysqld` a symlink to `mysqld_safe` in binary distribution.
- Fixed security bug when having an empty database name in the `user.db` table.
- Fixed some problems with `CREATE TABLE ... SELECT function()`.
- `mysqld` now has the option `--temp-pool` enabled by default as this gives better performance with some operating systems.
- Fixed hang in `CHANGE MASTER TO` if the slave thread died very quickly.
- Big cleanup in replication code (less logging, better error messages, etc..)
- If the `--code-file` option is specified, the server calls `setrlimit()` to set the maximum allowed core file size to unlimited, so core files can be generated.
- Fixed bug in query cache after temporary table creation.
- Added `--count=N` (`-c`) option to `mysqladmin`, to make the program do only `N` iterations. To be used with `--sleep` (`-i`). Useful in scripts.
- Fixed bug in multi-table `UPDATE`: when updating a table, `do_select()` became confused about reading records from a cache.
- Fixed bug in multi-table `UPDATE` when several fields were referenced from a single table
- Fixed bug in truncating nonexisting table.
- Fixed bug in `REVOKE` that caused user resources to be randomly set.
- Fixed bug in `GRANT` for the new `CREATE TEMPORARY TABLE` privilege.
- Fixed bug in multi-table `DELETE` when tables are re-ordered in the table initialisation method and ref_lengths are of different sizes.
- Fixed two bugs in `SELECT DISTINCT` with large tables.
- Fixed bug in query cache initialisation with very small query cache size.
- Allow `DEFAULT` with `INSERT` statement.
- The startup parameters `myisam_max_sort_file_size` and `myisam_max_extra_sort_file_size` are now given in bytes, not megabytes.
- External system locking of `MyISAM/ISAM` files is now turned off by default. One can turn this on with `--external-locking`. (For most users this is never needed).
- Fixed core dump bug with `INSERT ... SET db_name.table_name.colname=''`.
- Fixed client hangup bug when using some SQL commands with wrong syntax.
- Fixed a timing bug in `DROP DATABASE`
- New `SET [GLOBAL | SESSION]` syntax to change thread-specific and global server variables at runtime.
- Added variable `slave_compressed_protocol`.

- Renamed variable `query_cache_startup_type` to `query_cache_type`, `myisam_bulk_insert_tree_size` to `bulk_insert_buffer_size`, `record_buffer` to `read_buffer_size` and `record_rnd_buffer` to `record_rnd_buffer_size`.
- Renamed some SQL variables, but old names will still work until 5.0. Voir Section 2.5.1 [Upgrading-from-3.23], page 106.
- Renamed `--skip-locking` to `--skip-external-locking`.
- Removed unused variable `query_buffer_size`.
- Fixed a bug that made the pager option in the `mysql` client non-functional.
- Added full `AUTO_INCREMENT` support to `MERGE` tables.
- Extended `LOG()` function to accept an optional arbitrary base parameter. Voir Section 6.3.3.2 [Mathematical functions], page 456.
- Added `LOG2()` function (useful for finding out how many bits a number would require for storage).
- Added `LN()` natural logarithm function for compatibility with other databases. It is synonymous with `LOG(X)`.

D.2.5 Changes in release 4.0.2 (01 Jul 2002)

- Cleaned up `NULL` handling for default values in `DESCRIBE table_name`.
- Fixed `truncate()` to round up negative values to the nearest integer.
- Changed `--chroot=path` option to execute `chroot()` immediately after all options have been parsed.
- Don't allow database names that contain '\'.
- `lower_case_table_names` now also affects database names.
- Added `XOR` operator (logical and bitwise `XOR`) with `^` as a synonym for bitwise `XOR`.
- Added function `IS_FREE_LOCK("lock_name")`. Based on code contributed by Hartmut Holzgraefe `hartmut@six.de`.
- Removed `mysql_ssl_clear()` from C API, as it was not needed.
- `DECIMAL` and `NUMERIC` types can now read exponential numbers.
- Added `SHA1()` function to calculate 160 bit hash value as described in RFC 3174 (Secure Hash Algorithm). This function can be considered a cryptographically more secure equivalent of `MD5()`. Voir Section 6.3.6.2 [Miscellaneous functions], page 472.
- Added `AES_ENCRYPT()` and `AES_DECRYPT()` functions to perform encryption according to AES standard (Rijndael). Voir Section 6.3.6.2 [Miscellaneous functions], page 472.
- Added `--single-transaction` option to `mysqldump`, allowing a consistent dump of `InnoDB` tables. Voir Section 4.8.5 [mysqldump], page 318.
- Fixed bug in `innodb_log_group_home_dir` in `SHOW VARIABLES`.
- Fixed a bug in optimiser with merge tables when non-unique values are used in summing up (causing crashes).
- Fixed a bug in optimiser when a range specified makes index grouping impossible (causing crashes).
- Fixed a rare bug when `FULLTEXT` index is present and no tables are used.

- Added privileges `CREATE TEMPORARY TABLES`, `EXECUTE`, `LOCK TABLES`, `REPLICATION CLIENT`, `REPLICATION SLAVE`, `SHOW DATABASES` and `SUPER`. To use these, you must have run the `mysql_fix_privilege_tables` script after upgrading.
- Fixed query cache align data bug.
- Fixed mutex bug in replication when reading from master fails.
- Added missing mutex in `TRUNCATE TABLE`; This fixes some core dump/hangup problems when using `TRUNCATE TABLE`.
- Fixed bug in multi-table `DELETE` when optimiser uses only indices.
- Fixed that `ALTER TABLE table_name RENAME new_table_name` is as fast as `RENAME TABLE`.
- Fixed bug in `GROUP BY` with two or more fields, where at least one field can contain `NULL` values.
- Use `Turbo Boyer-Moore` algorithm to speed up `LIKE "%keyword%"` searches.
- Fixed bug in `DROP DATABASE` with symlink.
- Fixed crash in `REPAIR ... USE_FRM`.
- Fixed bug in `EXPLAIN` with `LIMIT offset != 0`.
- Fixed bug in phrase operator `"..."` in boolean full-text search.
- Fixed bug that caused duplicated rows when using truncation operator `*` in boolean full-text search.
- Fixed bug in truncation operator of boolean full-text search (wrong results when there are only `+word*`s in the query).
- Fixed bug in boolean full-text search that caused a crash when an identical `MATCH` expression that did not use an index appeared twice.
- Query cache is now automatically disabled in `mysqldump`.
- Fixed problem on Windows 98 that made sending of results very slow.
- Boolean full-text search weighting scheme changed to something more reasonable.
- Fixed bug in boolean full-text search that caused MySQL to ignore queries of `ft_min_word_len` characters.
- Boolean full-text search now supports “phrase searches”.
- New configure option `--without-query-cache`.
- Memory allocation strategy for “root memory” changed. Block size now grows with number of allocated blocks.
- `INET_NTOA()` now returns `NULL` if you give it an argument that is too large (greater than the value corresponding to `255.255.255.255`).
- Fix `SQL_CALC_FOUND_ROWS` to work with `UNION`s. It will work only if the first `SELECT` has this option and if there is global `LIMIT` for the entire statement. For the moment, this requires using parentheses for individual `SELECT` queries within the statement.
- Fixed bug in `SQL_CALC_FOUND_ROWS` and `LIMIT`.
- Don’t give an error for `CREATE TABLE ...(... VARCHAR(0))`.
- Fixed `SIGINT` and `SIGQUIT` problems in ‘`mysql.cc`’ on Linux with some `glibc` versions.
- Fixed bug in ‘`convert.cc`’, which is caused by having an incorrect `net_store_length()` linked in the `CONVERT::store()` method.

- DOUBLE and FLOAT columns now honor the UNSIGNED flag on storage.
- InnoDB now retains foreign key constraints through ALTER TABLE and CREATE/DROP INDEX.
- InnoDB now allows foreign key constraints to be added through the ALTER TABLE syntax.
- InnoDB tables can now be set to automatically grow in size (autoextend).
- Added --ignore-lines=n option to mysqlimport. This has the same effect as the IGNORE n LINES clause for LOAD DATA.
- Fixed bug in UNION with last offset being transposed to total result set.
- REPAIR ... USE_FRM added.
- Fixed that DEFAULT_SELECT_LIMIT is always imposed on UNION result set.
- Fixed that some SELECT options can appear only in the first SELECT.
- Fixed bug with LIMIT with UNION, where last select is in the braces.
- Fixed that full-text works fine with UNION operations.
- Fixed bug with indexless boolean full-text search.
- Fixed bug that sometimes appeared when full-text search was used with "const" tables.
- Fixed incorrect error value when doing a SELECT with an empty HEAP table.
- Use ORDER BY column DESC now sorts NULL values first. (In other words, NULL values sort first in all cases, whether or not DESC is specified.)
- Fixed bug in WHERE key_name='constant' ORDER BY key_name DESC.
- Fixed bug in SELECT DISTINCT ... ORDER BY DESC optimisation.
- Fixed bug in ... HAVING 'GROUP_FUNCTION'(xxx) IS [NOT] NULL.
- Fixed bug in truncation operator for boolean full-text search.
- Allow value of --user=# option for mysqld to be specified as a numeric user ID.
- Fixed a bug where SQL_CALC_ROWS returned an incorrect value when used with one table and ORDER BY and with InnoDB tables.
- Fixed that SELECT 0 LIMIT 0 doesn't hang thread.
- Fixed some problems with USE/IGNORE INDEX when using many keys with the same start column.
- Don't use table scan with BerkeleyDB and InnoDB tables when we can use an index that covers the whole row.
- Optimised InnoDB sort-buffer handling to take less memory.
- Fixed bug in multi-table DELETE and InnoDB tables.
- Fixed problem with TRUNCATE and InnoDB tables that produced the error Can't execute the given command because you have active locked tables or an active transaction.
- Added NO_UNSIGNED_SUBTRACTION to the set of flags that may be specified with the --sql-mode option for mysqld. It disables unsigned arithmetic rules when it comes to subtraction. (This will make MySQL 4.0 behave more closely to 3.23 with UNSIGNED columns).
- The result returned for all bit functions (|, <<, ...) is now of type unsigned integer.

- Added detection of `nan` values in `MyISAM` to make it possible to repair tables with `nan` in float or double columns.
- Fixed new bug in `myisamchk` where it didn't correctly update number of "parts" in the `MyISAM` index file.
- Changed to use `autoconf` 2.52 (from `autoconf` 2.13).
- Fixed optimisation problem where the MySQL Server was in "preparing" state for a long time when selecting from an empty table which had contained a lot of rows.
- Fixed bug in complicated join with `const` tables. This fix also improves performance a bit when referring to another table from a `const` table.
- First pre-version of multi-table `UPDATE` statement.
- Fixed bug in multi-table `DELETE`.
- Fixed bug in `SELECT CONCAT(argument_list) ... GROUP BY 1`.
- `INSERT ... SELECT` did a full rollback in case of an error. Fixed so that we only roll back the last statement in the current transaction.
- Fixed bug with empty expression for boolean full-text search.
- Fixed core dump bug in updating full-text key from/to `NULL`.
- ODBC compatibility: Added `BIT_LENGTH()` function.
- Fixed core dump bug in `GROUP BY BINARY column`.
- Added support for `NULL` keys in `HEAP` tables.
- Use index for `ORDER BY` in queries of type: `SELECT * FROM t WHERE key_part1=1 ORDER BY key_part1 DESC,key_part2 DESC`
- Fixed bug in `FLUSH QUERY CACHE`.
- Added `CAST()` and `CONVERT()` functions. The `CAST` and `CONVERT` functions are nearly identical and mainly useful when you want to create a column with a specific type in a `CREATE ... SELECT` statement. For more information, read Section 6.3.5 [Cast Functions], page 470.
- `CREATE ... SELECT` on `DATE` and `TIME` functions now create columns of the expected type.
- Changed order in which keys are created in tables.
- Added new columns `Null` and `Index_type` to `SHOW INDEX` output.
- Added `--no-beep` and `--prompt` options to `mysql` command-line client.
- New feature: management of user resources.

```
GRANT ... WITH MAX_QUERIES_PER_HOUR N1
               MAX_UPDATES_PER_HOUR N2
               MAX_CONNECTIONS_PER_HOUR N3;
```

  Voir Section 4.3.6 [User resources], page 236.
- Added `mysql_secure_installation` to the '`scripts/`' directory.

D.2.6 Changes in release 4.0.1 (23 Dec 2001)

- Fixed bug when `HANDLER` was used with some unsupported table type.

- `mysqldump` now puts `ALTER TABLE tbl_name DISABLE KEYS` and `ALTER TABLE tbl_name ENABLE KEYS` in the sql dump.
- Added `mysql_fix_extensions` script.
- Fixed stack overrun problem with `LOAD DATA FROM MASTER` on OSF/1.
- Fixed shutdown problem on HP-UX.
- Added `DES_ENCRYPT()` and `DES_DECRYPT()` functions.
- Added `FLUSH DES_KEY_FILE` statement.
- Added `--des-key-file` option to `mysqld`.
- `HEX(string)` now returns the characters in `string` converted to hexadecimal.
- Fixed problem with `GRANT` when using `lower_case_table_names=1`.
- Changed `SELECT ... IN SHARE MODE` to `SELECT ... LOCK IN SHARE MODE` (as in MySQL 3.23).
- A new query cache to cache results from identical `SELECT` queries.
- Fixed core dump bug on 64-bit machines when it got an incorrect communication packet.
- `MATCH ... AGAINST(... IN BOOLEAN MODE)` can now work without `FULLTEXT` index.
- Fixed slave to replicate from 3.23 master.
- Miscellaneous replication fixes/cleanup.
- Got shutdown to work on Mac OS X.
- Added '`myisam/ft_dump`' utility for low-level inspection of `FULLTEXT` indexes.
- Fixed bug in `DELETE ... WHERE ... MATCH ...`.
- Added support for `MATCH ... AGAINST(... IN BOOLEAN MODE)`. **Note: you must rebuild your tables with** `ALTER TABLE tablename TYPE=MyISAM` **to be able to use boolean full-text search**.
- `LOCATE()` and `INSTR()` are now case-sensitive if either argument is a binary string.
- Changed `RAND()` initialisation so that `RAND(N)` and `RAND(N+1)` are more distinct.
- Fixed core dump bug in `UPDATE ... ORDER BY`.
- Changed `INSERT INTO ... SELECT` to stop on errors by default.
- Ignore `DATA DIRECTORY` and `INDEX DIRECTORY` directives on Windows.
- Added boolean full-text search code. It should be considered early alpha.
- Extended `MODIFY` and `CHANGE` in `ALTER TABLE` to accept the `FIRST` and `AFTER` keywords.
- Indexes are now used with `ORDER BY` on a whole `InnoDB` table.

D.2.7 Changes in release 4.0.0 (Oct 2001: Alpha)

- Added `--xml` option to `mysql` for producing XML output.
- Added full-text variables `ft_min_word_len`, `ft_max_word_len`, and `ft_max_word_len_for_sort`.
- Added documentation for `libmysqld`, the embedded MySQL server library. Also added example programs (a `mysql` client and `mysqltest` test program) which use `libmysqld`.

- Removed all Gemini hooks from MySQL server.
- Removed `my_thread_init()` and `my_thread_end()` from '`mysql_com.h`', and added `mysql_thread_init()` and `mysql_thread_end()` to '`mysql.h`'.
- Support for communication packets > 16M. In 4.0.1 we will extend `MyISAM` to be able to handle these.
- Secure connections (with SSL).
- Unsigned `BIGINT` constants now work. `MIN()` and `MAX()` now handle signed and unsigned `BIGINT` numbers correctly.
- New character set `latin1_de` which provides correct German sorting.
- `STRCMP()` now uses the current character set when doing comparisons, which means that the default comparison behaviour now is case-insensitive.
- `TRUNCATE TABLE` and `DELETE FROM tbl_name` are now separate functions. One bonus is that `DELETE FROM tbl_name` now returns the number of deleted rows, rather than zero.
- `DROP DATABASE` now executes a `DROP TABLE` on all tables in the database, which fixes a problem with `InnoDB` tables.
- Added support for `UNION`.
- Added support for multi-table `DELETE` operations.
- A new `HANDLER` interface to `MyISAM` tables.
- Added support for `INSERT` on `MERGE` tables. Patch from Benjamin Pflugmann.
- Changed `WEEK(#,0)` to match the calendar in the USA.
- `COUNT(DISTINCT)` is about 30% faster.
- Speed up all internal list handling.
- Speed up `IS NULL`, `ISNULL()` and some other internal primitives.
- Full-text index creation now is much faster.
- Tree-like cache to speed up bulk inserts and `myisam_bulk_insert_tree_size` variable.
- Searching on packed (`CHAR`/`VARCHAR`) keys is now much faster.
- Optimised queries of type: `SELECT DISTINCT * from tbl_name ORDER by key_part1 LIMIT #`.
- `SHOW CREATE TABLE` now shows all table attributes.
- `ORDER BY ... DESC` can now use keys.
- `LOAD DATA FROM MASTER` "auto-magically" sets up a slave.
- Renamed `safe_mysqld` to `mysqld_safe` to make this name more in line with other MySQL scripts/commands.
- Added support for symbolic links to `MyISAM` tables. Symlink handling is now enabled by default for Windows.
- Added `SQL_CALC_FOUND_ROWS` and `FOUND_ROWS()`. This makes it possible to know how many rows a query would have returned without a `LIMIT` clause.
- Changed output format of `SHOW OPEN TABLES`.
- Allow `SELECT expression LIMIT ...`.
- Added `IDENTITY` as a synonym for `AUTO_INCREMENT` (like Sybase).

- Added `ORDER BY` syntax to `UPDATE` and `DELETE`.
- `SHOW INDEXES` is now a synonym for `SHOW INDEX`.
- Added `ALTER TABLE tbl_name DISABLE KEYS` and `ALTER TABLE tbl_name ENABLE KEYS` commands.
- Allow use of `IN` as a synonym for `FROM` in `SHOW` commands.
- Implemented "repair by sort" for `FULLTEXT` indexes. `REPAIR TABLE`, `ALTER TABLE`, and `OPTIMIZE TABLE` for tables with `FULLTEXT` indexes are now up to 100 times faster.
- Allow ANSI SQL syntax `X'hexadecimal-number'`.
- Cleaned up global lock handling for `FLUSH TABLES WITH READ LOCK`.
- Fixed problem with `DATETIME = constant` in `WHERE` optimisation.
- Added `--master-data` and `--no-autocommit` options to `mysqldump`. (Thanks to Brian Aker for this.)
- Added script `mysql_explain_log.sh` to distribution. (Thanks to mobile.de).

D.3 Changes in release 3.23.x (Stable)

The 3.23 release has several major features that are not present in previous versions. We have added three new table types:

`MyISAM` A new ISAM library which is tuned for SQL and supports large files.

`InnoDB` A transaction-safe table handler that supports row level locking, and many Oracle-like features.

`BerkeleyDB` or `BDB`
Uses the Berkeley DB library from Sleepycat Software to implement transaction-safe tables.

Note that only `MyISAM` is available in the standard binary distribution.

The 3.23 release also includes support for database replication between a master and many slaves, full-text indexing, and much more.

All new features are being developed in the 4.x version. Only bug fixes and minor enhancements to existing features will be added to 3.23.

The replication code and BerkeleyDB code is still not as tested and as the rest of the code, so we will probably need to do a couple of future releases of 3.23 with small fixes for this part of the code. As long as you don't use these features, you should be quite safe with MySQL 3.23!

Note that the above doesn't mean that replication or Berkeley DB don't work. We have done a lot of testing of all code, including replication and `BDB` without finding any problems. It only means that not as many users use this code as the rest of the code and because of this we are not yet 100% confident in this code.

D.3.1 Changes in release 3.23.54

- Fixed a bug that made `OPTIMIZE TABLE` to corrupt the table under some rare circumstances.

- Fixed `mysqlcheck` so it can deal with table names containing dashes.
- Fixed the bug that caused `IGNORE INDEX` and `USE INDEX` sometimes to be ignored.
- Fixed rare core dump problem in complicated `GROUP BY` queries that didn't return any result.
- Fixed a bug where `MATCH ... AGAINST () >=0` was treated as if it was `>`.
- One can create `TEMPORARY MERGE` tables now.
- Fixed that `--core-file` works on Linux (at least on kernel 2.4.18).
- Fixed a problem with `BDB` and `ALTER TABLE`.
- Fixed reference to freed memory when doing complicated `GROUP BY ... ORDER BY` queries. Symptom was that `mysqld` died in function `send_fields`.
- Allocate heap rows in smaller blocks to get better memory usage.
- Fixed memory allocation bug when storing `BLOB` values in internal temporary tables used for some (unlikely) `GROUP BY` queries.
- Fixed a bug in key optimizing handling where the expression `WHERE column_name = key_column_name` was calculated as true for `NULL` values.
- Fixed core dump bug when doing `LEFT JOIN ... WHERE key_column=NULL`.
- Fixed `MyISAM` crash when using dynamic-row tables with huge numbers of packed fields.

D.3.2 Changes in release 3.23.53 (09 Oct 2002)

- Fixed crash when `SHOW INNODB STATUS` was used and `skip-innodb` was defined.
- Fixed possible memory corruption bug in binary log file handling when slave rotated the logs (only affected 3.23, not 4.0).
- Fixed problem in `LOCK TABLES` on windows when one connects to a database that contains upper case letters.
- Fixed that `--skip-show-databases` doesn't reset the `--port` option.
- Small fix in `safe_mysqld` for some shells.
- Fixed that `FLUSH STATUS` doesn't reset `Delayed_insert_threads`.
- Fixed core dump bug when using the `BINARY` cast on a `NULL` value.
- Fixed race condition when someone did a `GRANT` at the same time a new user logged in or did a `USE DATABASE`.
- Fixed bug in `ALTER TABLE` and `RENAME TABLE` when running with `-O lower_case_table_names=1` (typically on windows) when giving the table name in uppercase.
- Fixed that `-O lower_case_table_names=1` also converts database names to lower case.
- Fixed unlikely core dump with `SELECT ... ORDER BY ... LIMIT`.
- Changed `AND/OR` to report that they can return NULL. This fixes a bug in `GROUP BY` on `AND/OR` expressions that return `NULL`.
- Fixed a bug that `OPTIMIZE` of locked and modified MyISAM table, reported table corruption.
- Fixed a `BDB`-related `ALTER TABLE` bug with dropping a column and shutting down immediately thereafter.

- Fixed problem with `configure ... --localstatedir=...`.
- Fixed problem with `UNSIGNED BIGINT` on AIX (again).
- Fixed bug in pthread_mutex_trylock() on HPUX 11.0.
- Multithreaded stress tests for InnoDB.

D.3.3 Changes in release 3.23.52 (14 Aug 2002)

- Wrap `BEGIN/COMMIT` around transaction in the binary log. This makes replication honour transactions.
- Fixed security bug when having an empty database name in the `user.db` table.
- Changed initialisation of `RND()` to make it less predicatable.
- Fixed problem with `GROUP BY` on result with expression that created a `BLOB` field.
- Fixed problem with `GROUP BY` on columns that have `NULL` values. To solve this we now create an `MyISAM` temporary table when doing a `GROUP BY` on a possible `NULL` item. From MySQL 4.0.5 we can use in memory `HEAP` tables for this case.
- Fixed problem with privilege tables when downgrading from 4.0.2 to 3.23.
- Fixed thread bug in `SLAVE START`, `SLAVE STOP` and automatic repair of MyISAM tables that could cause table cache to be corrupted.
- Fixed possible thread related key-cache-corruption problem with `OPTIMIZE TABLE` and `REPAIR TABLE`.
- Added name of 'administrator command' logs.
- Fixed bug with creating an auto-increment value on second part of a `UNIQUE()` key where first part could contain NULL values.
- Don't write slave-timeout reconnects to the error log.
- Fixed bug with slave net read timeouting
- Fixed a core-dump bug with `MERGE` tables and `MAX()` function.
- Fixed bug in `ALTER TABLE` with `BDB` tables.
- Fixed bug when logging `LOAD DATA INFILE` to binary log with no active database.
- Fixed a bug in range optimiser (causing crashes).
- Fixed possible problem in replication when doing `DROP DATABASE` on a database with `InnoDB` tables.
- Fixed that `mysql_info()` returns 0 for 'Duplicates' when using `INSERT DELAYED IGNORE`.
- Added `-DHAVE_BROKEN_REALPATH` to the Mac OS X (darwin) compile options in '`configure.in`' to fix a failure under high load.

D.3.4 Changes in release 3.23.51 (31 May 2002)

- Fix bug with closing tags missing slash for `mysqldump` XML output.
- Remove end space from `ENUM` values. (This fixed a problem with `SHOW CREATE TABLE`.)
- Fixed bug in `CONCAT_WS()` that cut the result.

- Changed name of server variables `Com_show_master_stat` to `Com_show_master_status` and `Com_show_slave_stat` to `Com_show_slave_status`.
- Changed handling of `gethostbyname()` to make the client library thread-safe even if `gethostbyname_r` doesn't exist.
- Fixed core-dump problem when giving a wrong password string to `GRANT`.
- Fixed bug in `DROP DATABASE` with symlinked directory.
- Fixed optimisation problem with `DATETIME` and value outside `DATETIME` range.
- Removed Sleepycat's `BDB` doc files from the source tree, as they're not needed (MySQL covers `BDB` in its own documentation).
- Fixed MIT-pthreads to compile with `glibc` 2.2 (needed for `make dist`).
- Fixed the `FLOAT(X+1,X)` is not converted to `FLOAT(X+2,X)`. (This also affected `DECIMAL`, `DOUBLE` and `REAL` types)
- Fixed the result from `IF()` is case in-sensitive if the second and third arguments are case sensitive.
- Fixed core dump problem on OSF/1 in `gethostbyname_r`.
- Fixed that underflowed decimal fields are not zero filled.
- If we get an overflow when inserting `'+11111'` for `DECIMAL(5,0) UNSIGNED` columns, we will just drop the sign.
- Fixed optimisation bug with `ISNULL(expression_which_cannot_be_null)` and `ISNULL(constant_expression)`.
- Fixed host lookup bug in the `glibc` library that we used with the 3.23.50 Linux-x86 binaries.

D.3.5 Changes in release 3.23.50 (21 Apr 2002)

- Fixed buffer overflow problem if someone specified a too long datadir parameter to mysqld
- Add missing `<row>` tags for `mysqldump` XML output.
- Fixed problem with `crash-me` and `gcc` 3.0.4.
- Fixed that `@@unknown_variable` doesn't hang server.
- Added `@@VERSION` as a synonym for `VERSION()`.
- `SHOW VARIABLES LIKE 'xxx'` is now case-insensitive.
- Fixed timeout for `GET_LOCK()` on HP-UX with DCE threads.
- Fixed memory allocation bug in the glibc library used to build Linux binaries, which caused mysqld to die in 'free()'.
- Fixed `SIGINT` and `SIGQUIT` problems in `mysql`.
- Fixed bug in character table converts when used with big (> 64K) strings.
- `InnoDB` now retains foreign key constraints through `ALTER TABLE` and `CREATE/DROP INDEX`.
- `InnoDB` now allows foreign key constraints to be added through the `ALTER TABLE` syntax.
- `InnoDB` tables can now be set to automatically grow in size (autoextend).

- Our Linux RPMS and binaries are now compiled with `gcc` 3.0.4, which should make them a bit faster.
- Fixed some buffer overflow problems when reading startup parameters.
- Because of problems on shutdown we have now disabled named pipes on windows by default. One can enable named pipes by starting mysqld with `--enable-named-pipe`.
- Fixed bug when using `WHERE key_column = 'J' or key_column='j'`.
- Fixed core-dump bug when using `--log-bin` with `LOAD DATA INFILE` without an active database.
- Fixed bug in `RENAME TABLE` when used with `lower_case_table_names=1` (default on Windows).
- Fixed unlikely core-dump bug when using `DROP TABLE` on a table that was in use by a thread that also used queries on only temporary tables.
- Fixed problem with `SHOW CREATE TABLE` and `PRIMARY KEY` when using 32 indexes.
- Fixed that one can use `SET PASSWORD` for the anonymous user.
- Fixed core dump bug when reading client groups from option files using `mysql_options()`.
- Memory leak (16 bytes per every **corrupted** table) closed.
- Fixed binary builds to use `--enable-local-infile`.
- Update source to work with new version of `bison`.
- Updated shell scripts to now agree with new POSIX standard.
- Fixed bug where `DATE_FORMAT()` returned empty string when used with `GROUP BY`.

D.3.6 Changes in release 3.23.49

- Don't give warning for a statement that is only a comment; this is needed for `mysqldump --disable-keys` to work.
- Fixed unlikely caching bug when doing a join without keys. In this case the last used field for a table always returned `NULL`.
- Added options to make `LOAD DATA LOCAL INFILE` more secure.
- MySQL binary release 3.23.48 for Linux contained a new `glibc` library, which has serious problems under high load and RedHat 7.2. The 3.23.49 binary release doesn't have this problem.
- Fixed shutdown problem on NT.

D.3.7 Changes in release 3.23.48 (07 Feb 2002)

- Added `--xml` option to `mysqldump` for producing XML output.
- Changed to use `autoconf` 2.52 (from `autoconf` 2.13)
- Fixed bug in complicated join with `const` tables.
- Added internal safety checks for `InnoDB`.
- Some `InnoDB` variables were always shown in `SHOW VARIABLES` as `OFF` on high-byte-first systems (like SPARC).

- Fixed problem with one thread using an `InnoDB` table and another thread doing an `ALTER TABLE` on the same table. Before that, `mysqld` could crash with an assertion failure in '`row0row.c`', line 474.
- Tuned the `InnoDB` SQL optimiser to favor index searches more often over table scans.
- Fixed a performance problem with `InnoDB` tables when several large `SELECT` queries are run concurrently on a multiprocessor Linux computer. Large CPU-bound `SELECT` queries will now also generally run faster on all platforms.
- If MySQL binlogging is used, `InnoDB` now prints after crash recovery the latest MySQL binlog name and the offset `InnoDB` was able to recover to. This is useful, for example, when resynchronising a master and a slave database in replication.
- Added better error messages to help in installation problems of `InnoDB` tables.
- It is now possible to recover MySQL temporary tables that have become orphaned inside the `InnoDB` tablespace.
- `InnoDB` now prevents a `FOREIGN KEY` declaration where the signedness is not the same in the referencing and referenced integer columns.
- Calling `SHOW CREATE TABLE` or `SHOW TABLE STATUS` could cause memory corruption and make `mysqld` crash. Especially at risk was `mysqldump`, because it frequently calls `SHOW CREATE TABLE`.
- If inserts to several tables containing an `AUTO_INCREMENT` column were wrapped inside one `LOCK TABLES`, `InnoDB` asserted in '`lock0lock.c`'.
- In 3.23.47 we allowed several `NULL` values in a `UNIQUE` secondary index for an `InnoDB` table. But `CHECK TABLE` was not relaxed: it reports the table as corrupt. `CHECK TABLE` no longer complains in this situation.
- `SHOW GRANTS` now shows `REFERENCES` instead of `REFERENCE`.

D.3.8 Changes in release 3.23.47 (27 Dec 2001)

- Fixed bug when using the following construct: `SELECT ... WHERE key=@var_name OR key=@var_name2`
- Restrict `InnoDB` keys to 500 bytes.
- `InnoDB` now supports `NULL` in keys.
- Fixed shutdown problem on HP-UX. (Introduced in 3.23.46)
- Fixed core dump bug in replication when using `SELECT RELEASE_LOCK()`.
- Added new command: `DO expression,[expression]`
- Added `slave-skip-errors` option.
- Added statistics variables for all MySQL commands. (`SHOW STATUS` is now much longer.)
- Fixed default values for `InnoDB` tables.
- Fixed that `GROUP BY expr DESC` works.
- Fixed bug when using `t1 LEFT JOIN t2 ON t2.key=constant`.
- `mysql_config` now also works with binary (relocated) distributions.

D.3.9 Changes in release 3.23.46 (29 Nov 2001)

- Fixed problem with aliased temporary table replication.
- `InnoDB` and `BDB` tables will now use index when doing an `ORDER BY` on the whole table.
- Fixed bug where one got an empty set instead of a DEADLOCK error when using `BDB` tables.
- One can now kill `ANALYZE`, `REPAIR`, and `OPTIMIZE TABLE` when the thread is waiting to get a lock on the table.
- Fixed race condition in `ANALYZE TABLE`.
- Fixed bug when joining with caching (unlikely to happen).
- Fixed race condition when using the binary log and `INSERT DELAYED` which could cause the binary log to have rows that were not yet written to `MyISAM` tables.
- Changed caching of binary log to make replication slightly faster.
- Fixed bug in replication on Mac OS X.

D.3.10 Changes in release 3.23.45 (22 Nov 2001)

- `(UPDATE|DELETE) ...WHERE MATCH` bugfix.
- shutdown should now work on Darwin (Mac OS X).
- Fixed core dump when repairing corrupted packed `MyISAM` files.
- `--core-file` now works on Solaris.
- Fix a bug which could cause `InnoDB` to complain if it cannot find free blocks from the buffer cache during recovery.
- Fixed bug in `InnoDB` insert buffer B-tree handling that could cause crashes.
- Fixed bug in `InnoDB` lock timeout handling.
- Fixed core dump bug in `ALTER TABLE` on a `TEMPORARY InnoDB` table.
- Fixed bug in `OPTIMIZE TABLE` that reset index cardinality if it was up to date.
- Fixed problem with `t1 LEFT_JOIN t2 ... WHERE t2.date_column IS NULL` when date_column was declared as `NOT NULL`.
- Fixed bug with `BDB` tables and keys on `BLOB` columns.
- Fixed bug in `MERGE` tables on OS with 32-bit file pointers.
- Fixed bug in `TIME_TO_SEC()` when using negative values.

D.3.11 Changes in release 3.23.44 (31 Oct 2001)

- Fixed `Rows_examined` count in slow query log.
- Fixed bug when using a reference to an `AVG()` column in `HAVING`.
- Fixed that date functions that require correct dates, like `DAYOFYEAR(column)`, will return `NULL` for `0000-00-00` dates.
- Fixed bug in const-propagation when comparing columns of different types. (`SELECT * FROM date_col="2001-01-01" and date_col=time_col`)

- Fixed bug that caused error message `Can't write, because of unique constraint` with some `GROUP BY` queries.
- Fixed problem with `sjis` character strings used within quoted table names.
- Fixed core dump when using `CREATE ... FULLTEXT` keys with other table handlers than `MyISAM`.
- Don't use `signal()` on Windows because this appears to not be 100% reliable.
- Fixed bug when doing `WHERE col_name=NULL` on an indexed column that had `NULL` values.
- Fixed bug when doing `LEFT JOIN ... ON (col_name = constant) WHERE col_name = constant`.
- When using replications, aborted queries that contained `%` could cause a core dump.
- `TCP_NODELAY` was not used on some systems. (Speed problem.)
- Applied portability fixes for OS/2. (Patch by Yuri Dario.)

The following changes are for `InnoDB` tables:

- Add missing `InnoDB` variables to `SHOW VARIABLES`.
- Foreign keys checking is now done for `InnoDB` tables.
- `DROP DATABASE` now works also for `InnoDB` tables.
- `InnoDB` now supports datafiles and raw disk partitions bigger than 4 GB on those operating systems that have big files.
- `InnoDB` calculates better table cardinality estimates for the MySQL optimiser.
- Accent characters in the default character set `latin1` are ordered according to the MySQL ordering.

  Note: if you are using `latin1` and have inserted characters whose code is greater than 127 into an indexed `CHAR` column, you should run `CHECK TABLE` on your table when you upgrade to 3.23.44, and drop and reimport the table if `CHECK TABLE` reports an error!
- A new '`my.cnf`' parameter, `innodb_thread_concurrency`, helps in performance tuning in heavily concurrent environments.
- A new '`my.cnf`' parameter, `innodb_fast_shutdown`, speeds up server shutdown.
- A new '`my.cnf`' parameter, `innodb_force_recovery`, helps to save your data in case the disk image of the database becomes corrupt.
- `innodb_monitor` has been improved and a new `innodb_table_monitor` added.
- Increased maximum key length from 500 to 7000 bytes.
- Fixed a bug in replication of `AUTO_INCREMENT` columns with multiple-line inserts.
- Fixed a bug when the case of letters changes in an update of an indexed secondary column.
- Fixed a hang when there are > 24 datafiles.
- Fixed a crash when `MAX(col)` is selected from an empty table, and `col` is not the first column in a multi-column index.
- Fixed a bug in purge which could cause crashes.

D.3.12 Changes in release 3.23.43 (04 Oct 2001)

- Fixed a bug in `INSERT DELAYED` and `FLUSH TABLES` introduced in 3.23.42.
- Fixed unlikely bug, which returned non-matching rows, in `SELECT` with many tables and multi-column indexes and 'range' type.
- Fixed an unlikely core dump bug when doing `EXPLAIN SELECT` when using many tables and `ORDER BY`.
- Fixed bug in `LOAD DATA FROM MASTER` when using table with `CHECKSUM=1`.
- Added unique error message when one gets a DEADLOCK during a transaction with `BDB` tables.
- Fixed problem with `BDB` tables and `UNIQUE` columns defined as `NULL`.
- Fixed problem with `myisampack` when using pre-space filled `CHAR` columns.
- Applied patch from Yuri Dario for OS/2.
- Fixed bug in `--safe-user-create`.

D.3.13 Changes in release 3.23.42 (08 Sep 2001)

- Fixed problem when using `LOCK TABLES` and `BDB` tables.
- Fixed problem with `REPAIR TABLE` on `MyISAM` tables with row lengths in the range from 65517 to 65520 bytes.
- Fixed rare hang when doing `mysqladmin shutdown` when there was a lot of activity in other threads.
- Fixed problem with `INSERT DELAYED` where delay thread could be hanging on `upgrading locks` with no apparent reason.
- Fixed problem with `myisampack` and `BLOB`.
- Fixed problem when one edited '`.MRG`' tables by hand. (Patch from Benjamin Pflugmann).
- Enforce that all tables in a `MERGE` table come from the same database.
- Fixed bug with `LOAD DATA INFILE` and transactional tables.
- Fix bug when using `INSERT DELAYED` with wrong column definition.
- Fixed core dump during `REPAIR` of some particularly broken tables.
- Fixed bug in `InnoDB` and `AUTO_INCREMENT` columns.
- Fixed bug in `InnoDB` and `RENAME TABLE` columns.
- Fixed critical bug in `InnoDB` and `BLOB` columns. If you have used `BLOB` columns larger than 8000 bytes in an `InnoDB` table, it is necessary to dump the table with `mysqldump`, drop it and restore it from the dump.
- Applied large patch for OS/2 from Yuri Dario.
- Fixed problem with `InnoDB` when one could get the error `Can't execute the given command...` even when no transaction was active.
- Applied some minor fixes that concern Gemini.
- Use real arithmetic operations even in integer context if not all arguments are integers. (Fixes uncommon bug in some integer contexts).

- Don't force everything to lowercase on Windows. (To fix problem with Windows and `ALTER TABLE`). Now `--lower_case_names` also works on Unix.
- Fixed that automatic rollback is done when thread end doesn't lock other threads.

D.3.14 Changes in release 3.23.41 (11 Aug 2001)

- Added `--sql-mode=option[,option[,option]]` option to `mysqld`. Voir Section 4.1.1 [Command-line options], page 192.
- Fixed possible problem with `shutdown` on Solaris where the '`.pid`' file wasn't deleted.
- `InnoDB` now supports < 4 GB rows. The former limit was 8000 bytes.
- The `doublewrite` file flush method is used in `InnoDB`. It reduces the need for Unix `fsync()` calls to a fraction and improves performance on most Unix flavors.
- You can now use the `InnoDB` Monitor to print a lot of `InnoDB` state information, including locks, to the standard output. This is useful in performance tuning.
- Several bugs which could cause hangs in `InnoDB` have been fixed.
- Split `record_buffer` to `record_buffer` and `record_rnd_buffer`. To make things compatible to previous MySQL versions, if `record_rnd_buffer` is not set, then it takes the value of `record_buffer`.
- Fixed optimising bug in `ORDER BY` where some `ORDER BY` parts where wrongly removed.
- Fixed overflow bug with `ALTER TABLE` and `MERGE` tables.
- Added prototypes for `my_thread_init()` and `my_thread_end()` to '`mysql_com.h`'
- Added `--safe-user-create` option to `mysqld`.
- Fixed bug in `SELECT DISTINCT ... HAVING` that caused error message `Can't find record in #...`

D.3.15 Changes in release 3.23.40

- Fixed problem with `--low-priority-updates` and `INSERT` statements.
- Fixed bug in slave thread when under some rare circumstances it could get 22 bytes ahead on the offset in the master.
- Added `slave_net_timeout` for replication.
- Fixed problem with `UPDATE` and `BDB` tables.
- Fixed hard bug in `BDB` tables when using key parts.
- Fixed problem when using `GRANT FILE ON database.* ...`; previously we added the `DROP` privilege for the database.
- Fixed `DELETE FROM tbl_name ... LIMIT 0` and `UPDATE FROM tbl_name ... LIMIT 0`, which acted as though the `LIMIT` clause was not present (they deleted or updated all selected rows).
- `CHECK TABLE` now checks if an `AUTO_INCREMENT` column contains the value 0.
- Sending a `SIGHUP` to `mysqld` will now only flush the logs, not reset the replication.
- Fixed parser to allow floats of type `1.0e1` (no sign after `e`).
- Option `--force` to `myisamchk` now also updates states.

- Added option `--warnings` to `mysqld`. Now `mysqld` prints the error `Aborted connection` only if this option is used.
- Fixed problem with `SHOW CREATE TABLE` when you didn't have a `PRIMARY KEY`.
- Properly fixed the rename of `innodb_unix_file_flush_method` variable to `innodb_flush_method`.
- Fixed bug when converting `BIGINT UNSIGNED` to `DOUBLE`. This caused a problem when doing comparisons with `BIGINT` values outside of the signed range.
- Fixed bug in `BDB` tables when querying empty tables.
- Fixed a bug when using `COUNT(DISTINCT)` with `LEFT JOIN` and there weren't any matching rows.
- Removed all documentation referring to the `GEMINI` table type. `GEMINI` is not released under an `Open Source` license.

D.3.16 Changes in release 3.23.39 (12 Jun 2001)

- The `AUTO_INCREMENT` sequence wasn't reset when dropping and adding an `AUTO_INCREMENT` column.
- `CREATE ... SELECT` now creates non-unique indexes delayed.
- Fixed problem where `LOCK TABLES tbl_name READ` followed by `FLUSH TABLES` put an exclusive lock on the table.
- `REAL @variable` values were represented with only 2 digits when converted to strings.
- Fixed problem that client "hung" when `LOAD TABLE FROM MASTER` failed.
- `myisamchk --fast --force` will no longer repair tables that only had the open count wrong.
- Added functions to handle symbolic links to make life easier in 4.0.
- We are now using the `-lcma` thread library on HP-UX 10.20 so that MySQL will be more stable on HP-UX.
- Fixed problem with `IF()` and number of decimals in the result.
- Fixed date-part extraction functions to work with dates where day and/or month is 0.
- Extended argument length in option files from 256 to 512 chars.
- Fixed problem with shutdown when `INSERT DELAYED` was waiting for a `LOCK TABLE`.
- Fixed core dump bug in `InnoDB` when tablespace was full.
- Fixed problem with `MERGE` tables and big tables (> 4G) when using `ORDER BY`.

D.3.17 Changes in release 3.23.38 (09 May 2001)

- Fixed a bug when `SELECT` from `MERGE` table sometimes results in incorrectly ordered rows.
- Fixed a bug in `REPLACE()` when using the `ujis` character set.
- Applied Sleepycat `BDB` patches 3.2.9.1 and 3.2.9.2.
- Added `--skip-stack-trace` option to `mysqld`.
- `CREATE TEMPORARY` now works with `InnoDB` tables.

- InnoDB now promotes sub keys to whole keys.
- Added option CONCURRENT to LOAD DATA.
- Better error message when slave max_allowed_packet is too low to read a very long log event from the master.
- Fixed bug when too many rows where removed when using SELECT DISTINCT ... HAVING.
- SHOW CREATE TABLE now returns TEMPORARY for temporary tables.
- Added Rows_examined to slow query log.
- Fixed problems with function returning empty string when used together with a group function and a WHERE that didn't match any rows.
- New program mysqlcheck.
- Added database name to output for administrative commands like CHECK, REPAIR, OPTIMIZE.
- Lots of portability fixes for InnoDB.
- Changed optimiser so that queries like SELECT * FROM tbl_name,tbl_name2 ... ORDER BY key_part1 LIMIT # will use index on key_part1 instead of filesort.
- Fixed bug when doing LOCK TABLE to_table WRITE,...; INSERT INTO to_table... SELECT ... when to_table was empty.
- Fixed bug with LOCK TABLE and BDB tables.

D.3.18 Changes in release 3.23.37 (17 Apr 2001)

- Fixed a bug when using MATCH() in HAVING clause.
- Fixed a bug when using HEAP tables with LIKE.
- Added --mysql-version option to safe_mysqld
- Changed INNOBASE to InnoDB (because the INNOBASE name was already used). All configure options and mysqld start options now use innodb instead of innobase. This means that before upgrading to this version, you have to change any configuration files where you have used innobase options!
- Fixed bug when using indexes on CHAR(255) NULL columns.
- Slave thread will now be started even if master-host is not set, as long as server-id is set and valid 'master.info' is present.
- Partial updates (terminated with kill) are now logged with a special error code to the binary log. Slave will refuse to execute them if the error code indicates the update was terminated abnormally, and will have to be recovered with SET SQL_SLAVE_SKIP_COUNTER=1; SLAVE START after a manual sanity check/correction of data integrity.
- Fixed bug that erroneously logged a drop of internal temporary table on thread termination to the binary log – this bug affected replication.
- Fixed a bug in REGEXP on 64-bit machines.
- UPDATE and DELETE with WHERE unique_key_part IS NULL didn't update/delete all rows.
- Disabled INSERT DELAYED for tables that support transactions.

- Fixed bug when using date functions on `TEXT`/`BLOB` column with wrong date format.
- UDFs now also work on Windows. (Patch by Ralph Mason.)
- Fixed bug in `ALTER TABLE` and `LOAD DATA INFILE` that disabled key-sorting. These commands should now be faster in most cases.
- Fixed performance bug where reopened tables (tables that had been waiting for `FLUSH` or `REPAIR`) would not use indexes for the next query.
- Fixed problem with `ALTER TABLE` to `InnoDB` tables on FreeBSD.
- Added `mysqld` variables `myisam_max_sort_file_size` and `myisam_max_extra_sort_file_size`.
- Initialise signals early to avoid problem with signals in `InnoDB`.
- Applied patch for the `tis620` character set to make comparisons case-independent and to fix a bug in `LIKE` for this character set. **Note**: All tables that uses the `tis620` character set must be fixed with `myisamchk -r` or `REPAIR TABLE` !
- Added `--skip-safemalloc` option to `mysqld`.

D.3.19 Changes in release 3.23.36 (27 Mar 2001)

- Fixed a bug that allowed use of database names containing a '`.`' character. This fixes a serious security issue when `mysqld` is run as root.
- Fixed bug when thread creation failed (could happen when doing a **lot** of connections in a short time).
- Fixed some problems with `FLUSH TABLES` and `TEMPORARY` tables. (Problem with freeing the key cache and error `Can't reopen table...`.)
- Fixed a problem in `InnoDB` with other character sets than `latin1` and another problem when using many columns.
- Fixed bug that caused a core dump when using a very complex query involving `DISTINCT` and summary functions.
- Added `SET TRANSACTION ISOLATION LEVEL ...`
- Added `SELECT ... FOR UPDATE`.
- Fixed bug where the number of affected rows was not returned when MySQL was compiled without transaction support.
- Fixed a bug in `UPDATE` where keys weren't always used to find the rows to be updated.
- Fixed a bug in `CONCAT_WS()` where it returned incorrect results.
- Changed `CREATE ... SELECT` and `INSERT ... SELECT` to not allow concurrent inserts as this could make the binary log hard to repeat. (Concurrent inserts are enabled if you are not using the binary or update log.)
- Changed some macros to be able to use fast mutex with `glibc` 2.2.

D.3.20 Changes in release 3.23.35 (15 Mar 2001)

- Fixed newly introduced bug in `ORDER BY`.
- Fixed wrong define `CLIENT_TRANSACTIONS`.

- Fixed bug in `SHOW VARIABLES` when using `INNOBASE` tables.
- Setting and using user variables in `SELECT DISTINCT` didn't work.
- Tuned `SHOW ANALYZE` for small tables.
- Fixed handling of arguments in the benchmark script `run-all-tests`.

D.3.21 Changes in release 3.23.34a

- Added extra files to the distribution to allow `INNOBASE` support to be compiled.

D.3.22 Changes in release 3.23.34 (10 Mar 2001)

- Added the `INNOBASE` table handler and the `BDB` table handler to the MySQL source distribution.
- Updated the documentation about `GEMINI` tables.
- Fixed a bug in `INSERT DELAYED` that caused threads to hang when inserting `NULL` into an `AUTO_INCREMENT` column.
- Fixed a bug in `CHECK TABLE` / `REPAIR TABLE` that could cause a thread to hang.
- `REPLACE` will not replace a row that conflicts with an `AUTO_INCREMENT` generated key.
- `mysqld` now only sets `CLIENT_TRANSACTIONS` in `mysql->server_capabilities` if the server supports a transaction-safe handler.
- Fixed `LOAD DATA INFILE` to allow numeric values to be read into `ENUM` and `SET` columns.
- Improved error diagnostic for slave thread exit.
- Fixed bug in `ALTER TABLE ... ORDER BY`.
- Added `max_user_connections` variable to `mysqld`.
- Limit query length for replication by `max_allowed_packet`, not the arbitrary limit of 4 MB.
- Allow space around `=` in argument to `--set-variable`.
- Fixed problem in automatic repair that could leave some threads in state `Waiting for table`.
- `SHOW CREATE TABLE` now displays the `UNION()` for `MERGE` tables.
- `ALTER TABLE` now remembers the old `UNION()` definition.
- Fixed bug when replicating timestamps.
- Fixed bug in bidirectional replication.
- Fixed bug in the `BDB` table handler that occurred when using an index on multi-part key where a key part may be `NULL`.
- Fixed `MAX()` optimisation on sub-key for `BDB` tables.
- Fixed problem where garbage results were returned when using `BDB` tables and `BLOB` or `TEXT` fields when joining many tables.
- Fixed a problem with `BDB` tables and `TEXT` columns.
- Fixed bug when using a `BLOB` key where a const row wasn't found.

- Fixed that `mysqlbinlog` writes the timestamp value for each query. This ensures that one gets same values for date functions like `NOW()` when using `mysqlbinlog` to pipe the queries to another server.
- Allow `--skip-gemini`, `--skip-bdb`, and `--skip-innodb` options to be specified when invoking `mysqld`, even if these table handlers are not compiled in to `mysqld`.
- One can now do `GROUP BY ... DESC`.
- Fixed a deadlock in the `SET` code, when one ran `SET @foo=bar`, where `bar` is a column reference, an error was not properly generated.

D.3.23 Changes in release 3.23.33 (09 Feb 2001)

- Fixed DNS lookups not to use the same mutex as the hostname cache. This will enable known hosts to be quickly resolved even if a DNS lookup takes a long time.
- Added `--character-sets-dir` option to `myisampack`.
- Removed warnings when running `REPAIR TABLE ... EXTENDED`.
- Fixed a bug that caused a core dump when using `GROUP BY` on an alias, where the alias was the same as an existing column name.
- Added `SEQUENCE()` as an example UDF function.
- Changed `mysql_install_db` to use `BINARY` for `CHAR` columns in the privilege tables.
- Changed `TRUNCATE tbl_name` to `TRUNCATE TABLE tbl_name` to use the same syntax as Oracle. Until 4.0 we will also allow `TRUNCATE tbl_name` to not crash old code.
- Fixed "no found rows" bug in `MyISAM` tables when a `BLOB` was first part of a multi-part key.
- Fixed bug where `CASE` didn't work with `GROUP BY`.
- Added `--sort-recover` option to `myisamchk`.
- `myisamchk -S` and `OPTIMIZE TABLE` now work on Windows.
- Fixed bug when using `DISTINCT` on results from functions that referred to a group function, like:

```
SELECT a, DISTINCT SEC_TO_TIME(SUM(a))
FROM tbl_name GROUP BY a, b;
```

- Fixed buffer overrun in `libmysqlclient` library. Fixed bug in handling `STOP` event after `ROTATE` event in replication.
- Fixed another buffer overrun in `DROP DATABASE`.
- Added `Table_locks_immediate` and `Table_locks_waited` status variables.
- Fixed bug in replication that broke slave server start with existing '`master.info`'. This fixes a bug introduced in 3.23.32.
- Added `SET SQL_SLAVE_SKIP_COUNTER=n` command to recover from replication glitches without a full database copy.
- Added `max_binlog_size` variable; the binary log will be rotated automatically when the size crosses the limit.
- Added `Last_error`, `Last_errno`, and `Slave_skip_counter` variables to `SHOW SLAVE STATUS`.

- Fixed bug in `MASTER_POS_WAIT()` function.
- Execute core dump handler on `SIGILL`, and `SIGBUS` in addition to `SIGSEGV`.
- On x86 Linux, print the current query and thread (connection) id, if available, in the core dump handler.
- Fixed several timing bugs in the test suite.
- Extended `mysqltest` to take care of the timing issues in the test suite.
- `ALTER TABLE` can now be used to change the definition for a `MERGE` table.
- Fixed creation of `MERGE` tables on Windows.
- Portability fixes for OpenBSD and OS/2.
- Added `--temp-pool` option to `mysqld`. Using this option will cause most temporary files created to use a small set of names, rather than a unique name for each new file. This is to work around a problem in the Linux kernel dealing with creating a bunch of new files with different names. With the old behaviour, Linux seems to "leak" memory, as it's being allocated to the directory entry cache instead of the disk cache.

D.3.24 Changes in release 3.23.32 (22 Jan 2001: Stable)

- Changed code to get around compiler bug in Compaq C++ on OSF/1, that broke `BACKUP`, `RESTORE`, `CHECK`, `REPAIR`, and `ANALYZE TABLE`.
- Added option `FULL` to `SHOW COLUMNS`. Now we show the privilege list for the columns only if this option is given.
- Fixed bug in `SHOW LOGS` when there weren't any `BDB` logs.
- Fixed a timing problem in replication that could delay sending an update to the client until a new update was done.
- Don't convert field names when using `mysql_list_fields()`. This is to keep this code compatible with `SHOW FIELDS`.
- `MERGE` tables didn't work on Windows.
- Fixed problem with `SET PASSWORD=...` on Windows.
- Added missing '`my_config.h`' to RPM distribution.
- `TRIM("foo" from "foo")` didn't return an empty string.
- Added `--with-version-suffix` option to `configure`.
- Fixed core dump when client aborted connection without `mysql_close()`.
- Fixed a bug in `RESTORE TABLE` when trying to restore from a non-existent directory.
- Fixed a bug which caused a core dump on the slave when replicating `SET PASSWORD`.
- Added `MASTER_POS_WAIT()`.

D.3.25 Changes in release 3.23.31 (17 Jan 2001)

- The test suite now tests all reachable `BDB` interface code. During testing we found and fixed many errors in the interface code.
- Using `HAVING` on an empty table could produce one result row when it shouldn't.
- Fixed the MySQL RPM not to depend on Perl5 anymore.

- Fixed some problems with `HEAP` tables on Windows.
- `SHOW TABLE STATUS` didn't show correct average row length for tables larger than 4G.
- `CHECK TABLE ... EXTENDED` didn't check row links for fixed size tables.
- Added option `MEDIUM` to `CHECK TABLE`.
- Fixed problem when using `DECIMAL()` keys on negative numbers.
- `HOUR()` (and some other `TIME` functions) on a `CHAR` column always returned `NULL`.
- Fixed security bug in something (please upgrade if you are using an earlier MySQL 3.23 version).
- Fixed buffer overflow bug when writing a certain error message.
- Added usage of `setrlimit()` on Linux to get `-O --open-files-limit=#` to work on Linux.
- Added `bdb_version` variable to `mysqld`.
- Fixed bug when using expression of type:

  ```
  SELECT ... FROM t1 LEFT JOIN t2 ON (t1.a=t2.a) WHERE t1.a=t2.a
  ```

  In this case the test in the `WHERE` clause was wrongly optimised away.
- Fixed bug in `MyISAM` when deleting keys with possible `NULL` values, but the first key-column was not a prefix-compressed text column.
- Fixed `mysql.server` to read the `[mysql.server]` option file group rather than the `[mysql_server]` group.
- Fixed `safe_mysqld` and `mysql.server` to also read the `server` option section.
- Added `Threads_created` status variable to `mysqld`.

D.3.26 Changes in release 3.23.30 (04 Jan 2001)

- Added `SHOW OPEN TABLES` command.
- Fixed that `myisamdump` works against old `mysqld` servers.
- Fixed `myisamchk -k#` so that it works again.
- Fixed a problem with replication when the binary log file went over 2G on 32-bit systems.
- `LOCK TABLES` will now automatically start a new transaction.
- Changed `BDB` tables to not use internal subtransactions and reuse open files to get more speed.
- Added `--mysqld=#` option to `safe_mysqld`.
- Allow hex constants in the `--fields-*-by` and `--lines-terminated-by` options to `mysqldump` and `mysqlimport`. By Paul DuBois.
- Added `--safe-show-database` option to `mysqld`.
- Added `have_bdb`, `have_gemini`, `have_innobase`, `have_raid` and `have_openssl` to `SHOW VARIABLES` to make it easy to test for supported extensions.
- Added `--open-files-limit` option to `mysqld`.
- Changed `--open-files` option to `--open-files-limit` in `safe_mysqld`.
- Fixed a bug where some rows were not found with `HEAP` tables that had many keys.

- Fixed that `--bdb-no-sync` works.
- Changed `--bdb-recover` to `--bdb-no-recover` as recover should be on by default.
- Changed the default number of `BDB` locks to 10000.
- Fixed a bug from 3.23.29 when allocating the shared structure needed for `BDB` tables.
- Changed `mysqld_multi.sh` to use configure variables. Patch by Christopher McCrory.
- Added fixing of include files for Solaris 2.8.
- Fixed bug with `--skip-networking` on Debian Linux.
- Fixed problem that some temporary files where reported as having the name `UNOPENED` in error messages.
- Fixed bug when running two simultaneous `SHOW LOGS` queries.

D.3.27 Changes in release 3.23.29 (16 Dec 2000)

- Configure updates for Tru64, large file support, and better TCP wrapper support. By Albert Chin-A-Young.
- Fixed bug in `<=>` operator.
- Fixed bug in `REPLACE` with `BDB` tables.
- `LPAD()` and `RPAD()` will shorten the result string if it's longer than the length argument.
- Added `SHOW LOGS` command.
- Remove unused `BDB` logs on shutdown.
- When creating a table, put `PRIMARY` keys first, followed by `UNIQUE` keys.
- Fixed a bug in `UPDATE` involving multi-part keys where one specified all key parts both in the update and the `WHERE` part. In this case MySQL could try to update a record that didn't match the whole `WHERE` part.
- Changed drop table to first drop the tables and then the '`.frm`' file.
- Fixed a bug in the hostname cache which caused `mysqld` to report the hostname as `''` in some error messages.
- Fixed a bug with `HEAP` type tables; the variable `max_heap_table_size` wasn't used. Now either `MAX_ROWS` or `max_heap_table_size` can be used to limit the size of a `HEAP` type table.
- Changed the default server-id to 1 for masters and 2 for slaves to make it easier to use the binary log.
- Renamed `bdb_lock_max` variable to `bdb_max_lock`.
- Added support for `AUTO_INCREMENT` on sub-fields for `BDB` tables.
- Added `ANALYZE` of `BDB` tables.
- In `BDB` tables, we now store the number of rows; this helps to optimise queries when we need an approximation of the number of rows.
- If we get an error in a multi-row statement, we now only roll back the last statement, not the entire transaction.
- If you do a `ROLLBACK` when you have updated a non-transactional table you will get an error as a warning.

- Added `--bdb-shared-data` option to `mysqld`.
- Added `Slave_open_temp_tables` status variable to `mysqld`
- Added `binlog_cache_size` and `max_binlog_cache_size` variables to `mysqld`.
- `DROP TABLE`, `RENAME TABLE`, `CREATE INDEX` and `DROP INDEX` are now transaction endpoints.
- If you do a `DROP DATABASE` on a symbolically linked database, both the link and the original database is deleted.
- Fixed `DROP DATABASE` to work on OS/2.
- Fixed bug when doing a `SELECT DISTINCT ... table1 LEFT JOIN table2 ...` when `table2` was empty.
- Added `--abort-slave-event-count` and `--disconnect-slave-event-count` options to `mysqld` for debugging and testing of replication.
- Fixed replication of temporary tables. Handles everything except slave server restart.
- `SHOW KEYS` now shows whether key is `FULLTEXT`.
- New script `mysqld_multi`. Voir Section 4.7.3 [mysqld_multi], page 293.
- Added new script, `mysql-multi.server.sh`. Thanks to Tim Bunce `Tim.Bunce@ig.co.uk` for modifying `mysql.server` to easily handle hosts running many `mysqld` processes.
- `safe_mysqld`, `mysql.server`, and `mysql_install_db` have been modified to use `mysql_print_defaults` instead of various hacks to read the '`my.cnf`' files. In addition, the handling of various paths has been made more consistent with how `mysqld` handles them by default.
- Automatically remove Berkeley DB transaction logs that no longer are in use.
- Fixed bug with several `FULLTEXT` indexes in one table.
- Added a warning if number of rows changes on `REPAIR/OPTIMIZE`.
- Applied patches for OS/2 by `Yuri Dario`.
- `FLUSH TABLES tbl_name` didn't always flush the index tree to disk properly.
- `--bootstrap` is now run in a separate thread. This fixes a problem that caused `mysql_install_db` to core dump on some Linux machines.
- Changed `mi_create()` to use less stack space.
- Fixed bug with optimiser trying to over-optimise `MATCH()` when used with `UNIQUE` key.
- Changed `crash-me` and the MySQL benchmarks to also work with FrontBase.
- Allow `RESTRICT` and `CASCADE` after `DROP TABLE` to make porting easier.
- Reset status variable which could cause problem if one used `--slow-log`.
- Added `connect_timeout` variable to `mysql` and `mysqladmin`.
- Added `connect-timeout` as an alias for `timeout` for option files read by `mysql_options()`.

D.3.28 Changes in release 3.23.28 (22 Nov 2000: Gamma)

- Added new options `--pager[=...]`, `--no-pager`, `--tee=...` and `--no-tee` to the `mysql` client. The new corresponding interactive commands are `pager`, `nopager`, `tee`

and `notee`. Voir Section 4.8.2 [`mysql`], page 307, `mysql --help` and the interactive help for more information.

- Fixed crash when automatic repair of `MyISAM` table failed.
- Fixed a major performance bug in the table locking code when one constantly had a lot of `SELECT`, `UPDATE` and `INSERT` statements running. The symptom was that the `UPDATE` and `INSERT` queries were locked for a long time while new `SELECT` statements were executed before the updates.
- When reading `options_files` with `mysql_options()` the `return-found-rows` option was ignored.
- One can now specify `interactive-timeout` in the option file that is read by `mysql_options()`. This makes it possible to force programs that run for a long time (like `mysqlhotcopy`) to use the `interactive_timeout` time instead of the `wait_timeout` time.
- Added to the slow query log the time and the user name for each logged query. If you are using `--log-long-format` then also queries that do not use an index are logged, even if the query takes less than `long_query_time` seconds.
- Fixed a problem in `LEFT JOIN` which caused all columns in a reference table to be `NULL`.
- Fixed a problem when using `NATURAL JOIN` without keys.
- Fixed a bug when using a multi-part keys where the first part was of type `TEXT` or `BLOB`.
- `DROP` of temporary tables wasn't stored in the update/binary log.
- Fixed a bug where `SELECT DISTINCT * ... LIMIT #` only returned one row.
- Fixed a bug in the assembler code in `strstr()` for SPARC and cleaned up the '`global.h`' header file to avoid a problem with bad aliasing with the compiler submitted with RedHat 7.0. (Reported by Trond Eivind Glomsrød)
- The `--skip-networking` option now works properly on NT.
- Fixed a long outstanding bug in the `ISAM` tables when a row with a length of more than 65K was shortened by a single byte.
- Fixed a bug in `MyISAM` when running multiple updating processes on the same table.
- Allow one to use `FLUSH TABLE tbl_name`.
- Added `--replicate-ignore-table`, `--replicate-do-table`, `--replicate-wild-ignore-table`, and `--replicate-wild-do-table` options to `mysqld`.
- Changed all log files to use our own `IO_CACHE` mechanism instead of `FILE` to avoid OS problems when there are many files open.
- Added `--open-files` and `--timezone` options to `safe_mysqld`.
- Fixed a fatal bug in `CREATE TEMPORARY TABLE ... SELECT ...`.
- Fixed a problem with `CREATE TABLE ... SELECT NULL`.
- Added variables `large_file_support`,`net_read_timeout`, `net_write_timeout` and `query_buffer_size` to `SHOW VARIABLES`.
- Added status variables `created_tmp_files` and `sort_merge_passes` to `SHOW STATUS`.
- Fixed a bug where we didn't allow an index name after the `FOREIGN KEY` definition.
- Added `TRUNCATE table_name` as a synonym for `DELETE FROM table_name`.

- Fixed a bug in a BDB key compare function when comparing part keys.
- Added bdb_lock_max variable to mysqld.
- Added more tests to the benchmark suite.
- Fixed an overflow bug in the client code when using overly long database names.
- mysql_connect() now aborts on Linux if the server doesn't answer in timeout seconds.
- SLAVE START did not work if you started with --skip-slave-start and had not explicitly run CHANGE MASTER TO.
- Fixed the output of SHOW MASTER STATUS to be consistent with SHOW SLAVE STATUS. (It now has no directory in the log name.)
- Added PURGE MASTER LOGS TO.
- Added SHOW MASTER LOGS.
- Added --safemalloc-mem-limit option to mysqld to simulate memory shortage when compiled with the --with-debug=full option.
- Fixed several core dumps in out-of-memory conditions.
- SHOW SLAVE STATUS was using an uninitialised mutex if the slave had not been started yet.
- Fixed bug in ELT() and MAKE_SET() when the query used a temporary table.
- CHANGE MASTER TO without specifying MASTER_LOG_POS would set it to 0 instead of 4 and hit the magic number in the master binlog.
- ALTER TABLE ... ORDER BY ... syntax added. This will create the new table with the rows in a specific order.

D.3.29 Changes in release 3.23.27 (24 Oct 2000)

- Fixed a bug where the automatic repair of MyISAM tables sometimes failed when the datafile was corrupt.
- Fixed a bug in SHOW CREATE when using AUTO_INCREMENT columns.
- Changed BDB tables to use new compare function in Berkeley DB 3.2.3.
- You can now use Unix sockets with MIT-pthreads.
- Added the latin5 (turkish) character set.
- Small portability fixes.

D.3.30 Changes in release 3.23.26 (18 Oct 2000)

- Renamed FLUSH MASTER and FLUSH SLAVE to RESET MASTER and RESET SLAVE.
- Fixed <> to work properly with NULL.
- Fixed a problem with SUBSTRING_INDEX() and REPLACE(). (Patch by Alexander Igonitchev)
- Fix CREATE TEMPORARY TABLE IF NOT EXISTS not to produce an error if the table exists.
- If you don't create a PRIMARY KEY in a BDB table, a hidden PRIMARY KEY will be created.
- Added read-only-key optimisation to BDB tables.

- `LEFT JOIN` in some cases preferred a full table scan when there was no `WHERE` clause.
- When using `--log-slow-queries`, don't count the time waiting for a lock.
- Fixed bug in lock code on Windows which could cause the key cache to report that the key file was crashed even if it was okay.
- Automatic repair of `MyISAM` tables if you start `mysqld` with `--myisam-recover`.
- Removed the `TYPE=` keyword from `CHECK` and `REPAIR`. Allow `CHECK` options to be combined. (You can still use `TYPE=`, but this usage is deprecated.)
- Fixed mutex bug in the binary replication log – long update queries could be read only in part by the slave if it did it at the wrong time, which was not fatal, but resulted in a performance-degrading reconnect and a scary message in the error log.
- Changed the format of the binary log – added magic number, server version, binlog version. Added server id and query error code for each query event.
- Replication thread from the slave now will kill all the stale threads from the same server.
- Long replication user names were not being handled properly.
- Added `--replicate-rewrite-db` option to `mysqld`.
- Added `--skip-slave-start` option to `mysqld`.
- Updates that generated an error code (such as `INSERT INTO foo(some_key) values (1),(1)`) erroneously terminated the slave thread.
- Added optimisation of queries where `DISTINCT` is only used on columns from some of the tables.
- Allow floating-point numbers where there is no sign after the exponent (like `1e1`).
- `SHOW GRANTS` didn't always show all column grants.
- Added `--default-extra-file=#` option to all MySQL clients.
- Columns referenced in `INSERT` statements now are initialised properly.
- `UPDATE` didn't always work when used with a range on a timestamp that was part of the key that was used to find rows.
- Fixed a bug in `FULLTEXT` index when inserting a `NULL` column.
- Changed to use `mkstemp()` instead of `tempnam()`. Based on a patch from John Jones.

D.3.31 Changes in release 3.23.25 (29 Sep 2000)

- Fixed that `databasename` works as second argument to `mysqlhotcopy`.
- The values for the `UMASK` and `UMASK_DIR` environment variables now can be specified in octal by beginning the value with a zero.
- Added `RIGHT JOIN`. This makes `RIGHT` a reserved word.
- Added `@@IDENTITY` as a synonym for `LAST_INSERT_ID()`. (This is for MSSQL compatibility.)
- Fixed a bug in `myisamchk` and `REPAIR` when using `FULLTEXT` index.
- `LOAD DATA INFILE` now works with FIFOs. (Patch by Toni L. Harbaugh-Blackford.)
- `FLUSH LOGS` broke replication if you specified a log name with an explicit extension as the value of the `log-bin` option.

- Fixed a bug in `MyISAM` with packed multi-part keys.
- Fixed crash when using `CHECK TABLE` on Windows.
- Fixed a bug where `FULLTEXT` index always used the `koi8_ukr` character set.
- Fixed privilege checking for `CHECK TABLE`.
- The `MyISAM` repair/reindex code didn't use the `--tmpdir` option for its temporary files.
- Added `BACKUP TABLE` and `RESTORE TABLE`.
- Fixed core dump on `CHANGE MASTER TO` when the slave did not have the master to start with.
- Fixed incorrect `Time` in the processlist for `Connect` of the slave thread.
- The slave now logs when it connects to the master.
- Fixed a core dump bug when doing `FLUSH MASTER` if you didn't specify a filename argument to `--log-bin`.
- Added missing '`ha_berkeley.x`' files to the MySQL Windows distribution.
- Fixed some mutex bugs in the log code that could cause thread blocks if new log files couldn't be created.
- Added lock time and number of selected processed rows to slow query log.
- Added `--memlock` option to `mysqld` to lock `mysqld` in memory on systems with the `mlockall()` call (like in Solaris).
- `HEAP` tables didn't use keys properly. (Bug from 3.23.23.)
- Added better support for `MERGE` tables (keys, mapping, creation, documentation...). Voir Section 7.2 [MERGE], page 539.
- Fixed bug in `mysqldump` from 3.23 which caused some `CHAR` columns not to be quoted.
- Merged `analyze`, `check`, `optimize` and repair code.
- `OPTIMIZE TABLE` is now mapped to `REPAIR` with statistics and sorting of the index tree. This means that for the moment it only works on `MyISAM` tables.
- Added a pre-alloced block to root_malloc to get fewer mallocs.
- Added a lot of new statistics variables.
- Fixed `ORDER BY` bug with `BDB` tables.
- Removed warning that `mysqld` couldn't remove the '`.pid`' file under Windows.
- Changed `--log-isam` to log `MyISAM` tables instead of isam tables.
- Fixed `CHECK TABLE` to work on Windows.
- Added file mutexes to make `pwrite()` safe on Windows.

D.3.32 Changes in release 3.23.24 (08 Sep 2000)

- Added `created_tmp_disk_tables` variable to `mysqld`.
- To make it possible to reliably dump and restore tables with `TIMESTAMP(X)` columns, MySQL now reports columns with `X` other than 14 or 8 to be strings.
- Changed sort order for `latin1` as it was before MySQL Version 3.23.23. Any table that was created or modified with 3.23.22 must be repaired if it has `CHAR` columns that may contain characters with ASCII values greater than 128!

- Fixed small memory leak introduced from 3.23.22 when creating a temporary table.
- Fixed problem with `BDB` tables and reading on a unique (not primary) key.
- Restored the `win1251` character set (it's now only marked deprecated).

D.3.33 Changes in release 3.23.23 (01 Sep 2000)

- Changed sort order for 'German'; all tables created with 'German' sortorder must be repaired with `REPAIR TABLE` or `myisamchk` before use!
- Added `--core-file` option to `mysqld` to get a core file on Linux if `mysqld` dies on the `SIGSEGV` signal.
- MySQL client `mysql` now starts with option `--no-named-commands` (`-g`) by default. This option can be disabled with `--enable-named-commands` (`-G`). This may cause incompatibility problems in some cases, for example, in SQL scripts that use named commands without a semicolon, etc. ! Long format commands still work from the first line.
- Fixed a problem when using many pending `DROP TABLE` statements at the same time.
- Optimiser didn't use keys properly when using `LEFT JOIN` on an empty table.
- Added shorter help text when invoking `mysqld` with incorrect options.
- Fixed non-fatal `free()` bug in `mysqlimport`.
- Fixed bug in `MyISAM` index handling of `DECIMAL/NUMERIC` keys.
- Fixed a bug in concurrent insert in `MyISAM` tables. In some contexts, usage of `MIN(key_part)` or `MAX(key_part)` returned an empty set.
- Updated `mysqlhotcopy` to use the new `FLUSH TABLES table_list` syntax. Only tables which are being backed up are flushed now.
- Changed behaviour of `--enable-thread-safe-client` so that both non-threaded (`-lmysqlclient`) and threaded (`-lmysqlclient_r`) libraries are built. Users who linked against a threaded `-lmysqlclient` will need to link against `-lmysqlclient_r` now.
- Added atomic `RENAME TABLE` command.
- Don't count `NULL` values in `COUNT(DISTINCT ...)`.
- Changed `ALTER TABLE`, `LOAD DATA INFILE` on empty tables and `INSERT ... SELECT ...` on empty tables to create non-unique indexes in a separate batch with sorting. This will make the above calls much faster when you have many indexes.
- `ALTER TABLE` now logs the first used insert_id correctly.
- Fixed crash when adding a default value to a `BLOB` column.
- Fixed a bug with `DATE_ADD/DATE_SUB` where it returned a datetime instead of a date.
- Fixed a problem with the thread cache which made some threads show up as `***DEAD***` in `SHOW PROCESSLIST`.
- Fixed a lock in our thr_rwlock code, which could make selects that run at the same time as concurrent inserts crash. This only affects systems that don't have the `pthread_rwlock_rdlock` code.
- When deleting rows with a non-unique key in a `HEAP` table, all rows weren't always deleted.

- Fixed bug in range optimiser for `HEAP` tables for searches on a part index.
- Fixed `SELECT` on part keys to work with `BDB` tables.
- Fixed `INSERT INTO bdb_table ... SELECT` to work with `BDB` tables.
- `CHECK TABLE` now updates key statistics for the table.
- `ANALYZE TABLE` will now only update tables that have been changed since the last `ANALYZE`. Note that this is a new feature and tables will not be marked to be analysed until they are updated in any way with 3.23.23 or newer. For older tables, you have to do `CHECK TABLE` to update the key distribution.
- Fixed some minor privilege problems with `CHECK`, `ANALYZE`, `REPAIR` and `SHOW CREATE` commands.
- Added `CHANGE MASTER TO` statement.
- Added `FAST`, `QUICK EXTENDED` check types to `CHECK TABLES`.
- Changed `myisamchk` so that `--fast` and `--check-only-changed` are also honored with `--sort-index` and `--analyze`.
- Fixed fatal bug in `LOAD TABLE FROM MASTER` that did not lock the table during index re-build.
- `LOAD DATA INFILE` broke replication if the database was excluded from replication.
- More variables in `SHOW SLAVE STATUS` and `SHOW MASTER STATUS`.
- `SLAVE STOP` now will not return until the slave thread actually exits.
- Full-text search via the `MATCH()` function and `FULLTEXT` index type (for `MyISAM` files). This makes `FULLTEXT` a reserved word.

D.3.34 Changes in release 3.23.22 (31 Jul 2000)

- Fixed that `lex_hash.h` is created properly for each MySQL distribution.
- Fixed that `MASTER` and `COLLECTION` are not reserved words.
- The log generated by `--slow-query-log` didn't contain the whole queries.
- Fixed that open transactions in `BDB` tables are rolled back if the connection is closed unexpectedly.
- Added workaround for a bug in `gcc` 2.96 (intel) and `gcc` 2.9 (IA64) in `gen_lex_hash.c`.
- Fixed memory leak in the client library when using `host=` in the '`my.cnf`' file.
- Optimised functions that manipulate the hours/minutes/seconds.
- Fixed bug when comparing the result of `DATE_ADD()`/`DATE_SUB()` against a number.
- Changed the meaning of `-F, --fast` for `myisamchk`. Added `-C, --check-only-changed` option to `myisamchk`.
- Added `ANALYZE tbl_name` to update key statistics for tables.
- Changed binary items `0x...` to be regarded as integers by default.
- Fix for SCO and `SHOW PROCESSLIST`.
- Added `auto-rehash` on reconnect for the `mysql` client.
- Fixed a newly introduced bug in `MyISAM`, where the index file couldn't get bigger than 64M.
- Added `SHOW MASTER STATUS` and `SHOW SLAVE STATUS`.

D.3.35 Changes in release 3.23.21

- Added `mysql_character_set_name()` function to the MySQL C API.
- Made the update log ASCII 0 safe.
- Added the `mysql_config` script.
- Fixed problem when using `<` or `>` with a char column that was only partly indexed.
- One would get a core dump if the log file was not readable by the MySQL user.
- Changed `mysqladmin` to use `CREATE DATABASE` and `DROP DATABASE` statements instead of the old deprecated API calls.
- Fixed `chown` warning in `safe_mysqld`.
- Fixed a bug in `ORDER BY` that was introduced in 3.23.19.
- Only optimise the `DELETE FROM tbl_name` to do a drop+create of the table if we are in `AUTOCOMMIT` mode (needed for `BDB` tables).
- Added extra checks to avoid index corruption when the `ISAM`/`MyISAM` index files get full during an `INSERT`/`UPDATE`.
- `myisamchk` didn't correctly update row checksum when used with `-ro` (this only gave a warning in subsequent runs).
- Fixed bug in `REPAIR TABLE` so that it works with tables without indexes.
- Fixed buffer overrun in `DROP DATABASE`.
- `LOAD TABLE FROM MASTER` is sufficiently bug-free to announce it as a feature.
- `MATCH` and `AGAINST` are now reserved words.

D.3.36 Changes in release 3.23.20

- Fixed bug in 3.23.19; `DELETE FROM tbl_name` removed the '`.frm`' file.
- Added `SHOW CREATE TABLE`.

D.3.37 Changes in release 3.23.19

- Changed copyright for all files to GPL for the server code and utilities and to LGPL for the client libraries.
- Fixed bug where all rows matching weren't updated on a `MyISAM` table when doing update based on key on a table with many keys and some key changed values.
- The Linux MySQL RPM's and binaries are now statically linked with a linuxthread version that has faster mutex handling when used with MySQL.
- `ORDER BY` can now use `REF` keys to find subsets of the rows that need to be sorted.
- Changed name of `print_defaults` program to `my_print_defaults` to avoid name confusion.
- Fixed `NULLIF()` to work according to ANSI SQL99.
- Added `net_read_timeout` and `net_write_timeout` as startup parameters to `mysqld`.
- Fixed bug that destroyed index when doing `myisamchk --sort-records` on a table with prefix compressed index.

- Added `pack_isam` and `myisampack` to the standard MySQL distribution.
- Added the syntax `BEGIN WORK` (the same as `BEGIN`).
- Fixed core dump bug when using `ORDER BY` on a `CONV()` expression.
- Added `LOAD TABLE FROM MASTER`.
- Added `FLUSH MASTER` and `FLUSH SLAVE`.
- Fixed big/little endian problem in the replication.

D.3.38 Changes in release 3.23.18

- Fixed a problem from 3.23.17 when choosing character set on the client side.
- Added `FLUSH TABLES WITH READ LOCK` to make a global lock suitable for making a copy of MySQL datafiles.
- `CREATE TABLE ... SELECT ... PROCEDURE` now works.
- Internal temporary tables will now use compressed index when using `GROUP BY` on `VARCHAR/CHAR` columns.
- Fixed a problem when locking the same table with both a `READ` and a `WRITE` lock.
- Fixed problem with `myisamchk` and `RAID` tables.

D.3.39 Changes in release 3.23.17

- Fixed a bug in `FIND_IN_SET()` when the first argument was `NULL`.
- Added table locks to Berkeley DB.
- Fixed a bug with `LEFT JOIN` and `ORDER BY` where the first table had only one matching row.
- Added 4 sample '`my.cnf`' example files in the '`support-files`' directory.
- Fixed `duplicated key` problem when doing big `GROUP BY` operations. (This bug was probably introduced in 3.23.15.)
- Changed syntax for `INNER JOIN` to match ANSI SQL.
- Added `NATURAL JOIN` syntax.
- A lot of fixes in the `BDB` interface.
- Added handling of `--no-defaults` and `--defaults-file` to `safe_mysqld.sh` and `mysql_install_db.sh`.
- Fixed bug in reading compressed tables with many threads.
- Fixed that `USE INDEX` works with `PRIMARY` keys.
- Added `BEGIN` statement to start a transaction in `AUTOCOMMIT` mode.
- Added support for symbolic links for Windows.
- Changed protocol to let client know if the server is in `AUTOCOMMIT` mode and if there is a pending transaction. If there is a pending transaction, the client library will give an error before reconnecting to the server to let the client know that the server did a rollback. The protocol is still backward-compatible with old clients.
- `KILL` now works on a thread that is locked on a 'write' to a dead client.

- Fixed memory leak in the replication slave thread.
- Added new `log-slave-updates` option to `mysqld`, to allow daisy-chaining the slaves.
- Fixed compile error on FreeBSD and other systems where `pthread_t` is not the same as `int`.
- Fixed master shutdown aborting the slave thread.
- Fixed a race condition in `INSERT DELAYED` code when doing `ALTER TABLE`.
- Added deadlock detection sanity checks to `INSERT DELAYED`.

D.3.40 Changes in release 3.23.16

- Added `SLAVE START` and `SLAVE STOP` statements.
- Added `TYPE=QUICK` option to `CHECK` and to `REPAIR`.
- Fixed bug in `REPAIR TABLE` when the table was in use by other threads.
- Added a thread cache to make it possible to debug MySQL with `gdb` when one does a lot of reconnects. This will also improve systems where you can't use persistent connections.
- Lots of fixes in the Berkeley DB interface.
- `UPDATE IGNORE` will not abort if an update results in a `DUPLICATE_KEY` error.
- Put `CREATE TEMPORARY TABLE` commands in the update log.
- Fixed bug in handling of masked IP numbers in the privilege tables.
- Fixed bug with `delay_key_write` tables and `CHECK TABLE`.
- Added `replicate-do-db` and `replicate-ignore-db` options to `mysqld`, to restrict which databases get replicated.
- Added `SQL_LOG_BIN` option.

D.3.41 Changes in release 3.23.15 (May 2000: Beta)

- To start `mysqld` as `root`, you must now use the `--user=root` option.
- Added interface to Berkeley DB. (This is not yet functional; play with it at your own risk!)
- Replication between master and slaves.
- Fixed bug that other threads could steal a lock when a thread had a lock on a table and did a `FLUSH TABLES` command.
- Added the `slow_launch_time` variable and the `Slow_launch_threads` status variable to `mysqld`. These can be examined with `mysqladmin variables` and `mysqladmin extended-status`.
- Added functions `INET_NTOA()` and `INET_ATON()`.
- The default type of `IF()` now depends on the second and third arguments and not only on the second argument.
- Fixed case when `myisamchk` could go into a loop when trying to repair a crashed table.
- Don't write `INSERT DELAYED` to update log if `SQL_LOG_UPDATE=0`.
- Fixed problem with `REPLACE` on `HEAP` tables.

- Added possible character sets and time zone to `SHOW VARIABLES` output.
- Fixed bug in locking code that could result in locking problems with concurrent inserts under high load.
- Fixed a problem with `DELETE` of many rows on a table with compressed keys where MySQL scanned the index to find the rows.
- Fixed problem with `CHECK` on table with deleted keyblocks.
- Fixed a bug in reconnect (at the client side) where it didn't free memory properly in some contexts.
- Fixed problems in update log when using `LAST_INSERT_ID()` to update a table with an `AUTO_INCREMENT` key.
- Added `NULLIF()` function.
- Fixed bug when using `LOAD DATA INFILE` on a table with `BLOB/TEXT` columns.
- Optimised `MyISAM` to be faster when inserting keys in sorted order.
- `EXPLAIN SELECT ...` now also prints out whether MySQL needs to create a temporary table or use file sorting when resolving the `SELECT`.
- Added optimisation to skip `ORDER BY` parts where the part is a constant expression in the `WHERE` part. Indexes can now be used even if the `ORDER BY` doesn't match the index exactly, as long as all the unused index parts and all the extra `ORDER BY` columns are constants in the `WHERE` clause. Voir Section 5.4.3 [MySQL indexes], page 384.
- `UPDATE` and `DELETE` on a whole unique key in the `WHERE` part are now faster than before.
- Changed `RAID_CHUNKSIZE` to be in 1024-byte increments.
- Fixed core dump in `LOAD_FILE(NULL)`.

D.3.42 Changes in release 3.23.14

- Added `mysql_real_escape_string()` function to the MySQL C API.
- Fixed a bug in `CONCAT()` where one of the arguments was a function that returned a modified argument.
- Fixed a critical bug in `myisamchk`, where it updated the header in the index file when one only checked the table. This confused the `mysqld` daemon if it updated the same table at the same time. Now the status in the index file is only updated if one uses `--update-state`. With older `myisamchk` versions you should use `--read-only` when only checking tables, if there is the slightest chance that the `mysqld` server is working on the table at the same time!
- Fixed that `DROP TABLE` is logged in the update log.
- Fixed problem when searching on `DECIMAL()` key field where the column data contained leading zeros.
- Fix bug in `myisamchk` when the `AUTO_INCREMENT` column isn't the first key.
- Allow `DATETIME` in ISO8601 format: 2000-03-12T12:00:00
- Dynamic character sets. A `mysqld` binary can now handle many different character sets (you can choose which when starting `mysqld`).
- Added command `REPAIR TABLE`.

- Added `mysql_thread_safe()` function to the MySQL C API.
- Added the `UMASK_DIR` environment variable.
- Added `CONNECTION_ID()` function to return the client connection thread ID.
- When using `=` on `BLOB` or `VARCHAR BINARY` keys, where only a part of the column was indexed, the whole column of the result row wasn't compared.
- Fix for `sjis` character set and `ORDER BY`.
- When running in ANSI mode, don't allow columns to be used that aren't in the `GROUP BY` part.

D.3.43 Changes in release 3.23.13

- Fixed problem when doing locks on the same table more than 2 times in the same `LOCK TABLE` command; this fixed the problem one got when running the test-ATIS test with `--fast` or `--check-only-changed`.
- Added `SQL_BUFFER_RESULT` option to `SELECT`.
- Removed end space from double/float numbers in results from temporary tables.
- Added `CHECK TABLE` command.
- Added changes for `MyISAM` in 3.23.12 that didn't get into the source distribution because of CVS problems.
- Fixed bug so that `mysqladmin shutdown` will wait for the local server to close down.
- Fixed a possible endless loop when calculating timestamp.
- Added `print_defaults` program to the '`.rpm`' files. Removed `mysqlbug` from the client '`.rpm`' file.

D.3.44 Changes in release 3.23.12 (07 Mar 2000)

- Fixed bug in `MyISAM` involving `REPLACE ... SELECT ...` which could give a corrupted table.
- Fixed bug in `myisamchk` where it incorrectly reset the `AUTO_INCREMENT` value.
- LOTS of patches for Linux Alpha. MySQL now appears to be relatively stable on Alpha.
- Changed `DISTINCT` on `HEAP` temporary tables to use hashed keys to quickly find duplicated rows. This mostly concerns queries of type `SELECT DISTINCT ... GROUP BY ...`. This fixes a problem where not all duplicates were removed in queries of the above type. In addition, the new code is MUCH faster.
- Added patches to make MySQL compile on Mac OS X.
- Added `IF NOT EXISTS` clause to `CREATE DATABASE`.
- Added `--all-databases` and `--databases` options to `mysqldump` to allow dumping of many databases at the same time.
- Fixed bug in compressed `DECIMAL()` index in `MyISAM` tables.
- Fixed bug when storing 0 into a timestamp.
- When doing `mysqladmin shutdown` on a local connection, `mysqladmin` now waits until the PID file is gone before terminating.

- Fixed core dump with some `COUNT(DISTINCT ...)` queries.
- Fixed that `myisamchk` works properly with RAID tables.
- Fixed problem with `LEFT JOIN` and `key_field IS NULL`.
- Fixed bug in `net_clear()` which could give the error `Aborted connection` in the MySQL clients.
- Added options `USE INDEX (key_list)` and `IGNORE INDEX (key_list)` as parameters in `SELECT`.
- `DELETE` and `RENAME` should now work on `RAID` tables.

D.3.45 Changes in release 3.23.11

- Allow the `ALTER TABLE tbl_name ADD (field_list)` syntax.
- Fixed problem with optimiser that could sometimes use incorrect keys.
- Fixed that `GRANT/REVOKE ALL PRIVILEGES` doesn't affect `GRANT OPTION`.
- Removed extra '`)`' from the output of `SHOW GRANTS`.
- Fixed problem when storing numbers in timestamps.
- Fix problem with timezones that have half hour offsets.
- Allow the syntax `UNIQUE INDEX` in `CREATE` statements.
- `mysqlhotcopy` - fast online hot-backup utility for local MySQL databases. By Tim Bunce.
- New more secure `mysqlaccess`. Thanks to Steve Harvey for this.
- Added `--i-am-a-dummy` and `--safe-updates` options to `mysql`.
- Added `select_limit` and `max_join_size` variables to `mysql`.
- Added `SQL_MAX_JOIN_SIZE` and `SQL_SAFE_UPDATES` options.
- Added `READ LOCAL` lock that doesn't lock the table for concurrent inserts. (This is used by `mysqldump`.)
- Changed that `LOCK TABLES ... READ` doesn't anymore allow concurrent inserts.
- Added `--skip-delay-key-write` option to `mysqld`.
- Fixed security problem in the protocol regarding password checking.
- `_rowid` can now be used as an alias for an integer type unique indexed column.
- Added back blocking of `SIGPIPE` when compiling with `--thread-safe-clients` to make things safe for old clients.

D.3.46 Changes in release 3.23.10

- Fixed bug in 3.23.9 where memory wasn't properly freed when using `LOCK TABLES`.

D.3.47 Changes in release 3.23.9

- Fixed problem that affected queries that did arithmetic on group functions.
- Fixed problem with timestamps and `INSERT DELAYED`.
- Fixed that `date_col BETWEEN const_date AND const_date` works.

- Fixed problem when only changing a 0 to `NULL` in a table with `BLOB/TEXT` columns.
- Fixed bug in range optimiser when using many key parts and or on the middle key parts: `WHERE K1=1 and K3=2 and (K2=2 and K4=4 or K2=3 and K4=5)`
- Added `source` command to `mysql` to allow reading of batch files inside the `mysql` client. Original patch by Matthew Vanecek.
- Fixed critical problem with the `WITH GRANT OPTION` option.
- Don't give an unnecessary `GRANT` error when using tables from many databases in the same query.
- Added VIO wrapper (needed for SSL support; by Andrei Errapart and Tõnu Samuel).
- Fixed optimiser problem on `SELECT` when using many overlapping indexes. MySQL should now be able to choose keys even better when there are many keys to choose from.
- Changed optimiser to prefer a range key instead of a ref key when the range key can uses more columns than the ref key (which only can use columns with `=`). For example, the following type of queries should now be faster: `SELECT * from key_part_1=const and key_part_2 > const2`
- Fixed bug that a change of all `VARCHAR` columns to `CHAR` columns didn't change row type from dynamic to fixed.
- Disabled floating-point exceptions for FreeBSD to fix core dump when doing `SELECT FLOOR(POW(2,63))`.
- Renamed `mysqld` startup option from `--delay-key-write` to `--delay-key-write-for-all-tables`.
- Added `read-next-on-key` to `HEAP` tables. This should fix all problems with `HEAP` tables when using non-`UNIQUE` keys.
- Added option to print default arguments to all clients.
- Added `--log-slow-queries` option to `mysqld` to log all queries that take a long time to a separate log file with a time indicating how long the query took.
- Fixed core dump when doing `WHERE key_col=RAND(...)`.
- Fixed optimisation bug in `SELECT ... LEFT JOIN ... key_col IS NULL`, when `key_col` could contain `NULL` values.
- Fixed problem with 8-bit characters as separators in `LOAD DATA INFILE`.

D.3.48 Changes in release 3.23.8 (02 Jan 2000)

- Fixed problem when handling indexfiles larger than 8G.
- Added latest patches to MIT-pthreads for NetBSD.
- Fixed problem with timezones that are < GMT - 11.
- Fixed a bug when deleting packed keys in `NISAM`.
- Fixed problem with `ISAM` when doing some `ORDER BY ... DESC` queries.
- Fixed bug when doing a join on a text key which didn't cover the whole key.
- Option `--delay-key-write` didn't enable delayed key writing.
- Fixed update of `TEXT` column which involved only case changes.

- Fixed that `INSERT DELAYED` doesn't update timestamps that are given.
- Added function `YEARWEEK()` and options `x`, `X`, `v` and `V` to `DATE_FORMAT()`.
- Fixed problem with `MAX(indexed_column)` and `HEAP` tables.
- Fixed problem with `BLOB NULL` keys and `LIKE "prefix%"`.
- Fixed problem with `MyISAM` and fixed-length rows < 5 bytes.
- Fixed problem that could cause MySQL to touch freed memory when doing very complicated `GROUP BY` queries.
- Fixed core dump if you got a crashed table where an `ENUM` field value was too big.

D.3.49 Changes in release 3.23.7 (10 Dec 1999)

- Fixed workaround under Linux to avoid problems with `pthread_mutex_timedwait`, which is used with `INSERT DELAYED`. Voir Section 2.6.1 [Linux], page 113.
- Fixed that one will get a 'disk full' error message if one gets disk full when doing sorting (instead of waiting until we got more disk space).
- Fixed a bug in `MyISAM` with keys > 250 characters.
- In `MyISAM` one can now do an `INSERT` at the same time as other threads are reading from the table.
- Added `max_write_lock_count` variable to `mysqld` to force a `READ` lock after a certain number of `WRITE` locks.
- Inverted flag `delay_key_write` on `show variables`.
- Renamed `concurrency` variable to `thread_concurrency`.
- The following functions are now multi-byte-safe: `LOCATE(substr,str)`, `POSITION(substr IN str)`, `LOCATE(substr,str,pos)`, `INSTR(str,substr)`, `LEFT(str,len)`, `RIGHT(str,len)`, `SUBSTRING(str,pos,len)`, `SUBSTRING(str FROM pos FOR len)`, `MID(str,pos,len)`, `SUBSTRING(str,pos)`, `SUBSTRING(str FROM pos)`, `SUBSTRING_INDEX(str,delim,count)`, `RTRIM(str)`, `TRIM([[BOTH | TRAILING] [remstr] FROM] str)`, `REPLACE(str,from_str,to_str)`, `REVERSE(str)`, `INSERT(str,pos,len,newstr)`, `LCASE(str)`, `LOWER(str)`, `UCASE(str)` and `UPPER(str)`; patch by Wei He.
- Fix core dump when releasing a lock from a non-existent table.
- Remove locks on tables before starting to remove duplicates.
- Added option `FULL` to `SHOW PROCESSLIST`.
- Added option `--verbose` to `mysqladmin`.
- Fixed problem when automatically converting `HEAP` to `MyISAM`.
- Fixed bug in `HEAP` tables when doing insert + delete + insert + scan the table.
- Fixed bugs on Alpha with `REPLACE()` and `LOAD DATA INFILE`.
- Added `interactive_timeout` variable to `mysqld`.
- Changed the argument to `mysql_data_seek()` from `ulong` to `ulonglong`.

D.3.50 Changes in release 3.23.6

- Added `-O lower_case_table_names={0|1}` option to `mysqld` to allow users to force table names to lowercase.
- Added `SELECT ... INTO DUMPFILE`.
- Added `--ansi` option to `mysqld` to make some functions ANSI SQL compatible.
- Temporary table names now start with `#sql`.
- Added quoting of identifiers with ‘ (`"` in `--ansi` mode).
- Changed to use `snprintf()` when printing floats to avoid some buffer overflows on FreeBSD.
- Made `FLOOR()` overflow safe on FreeBSD.
- Added `--quote-names` option to `mysqldump`.
- Fixed bug that one could make a part of a `PRIMARY KEY NOT NULL`.
- Fixed `encrypt()` to be thread-safe and not reuse buffer.
- Added `mysql_odbc_escape_string()` function to support big5 characters in MyODBC.
- Rewrote the table handler to use classes. This introduces a lot of new code, but will make table handling faster and better.
- Added patch by Sasha for user-defined variables.
- Changed that `FLOAT` and `DOUBLE` (without any length modifiers) no longer are fixed decimal point numbers.
- Changed the meaning of `FLOAT(X)`: Now this is the same as `FLOAT` if `X` <= 24 and a `DOUBLE` if 24 < `X` <= 53.
- `DECIMAL(X)` is now an alias for `DECIMAL(X,0)` and `DECIMAL` is now an alias for `DECIMAL(10,0)`. The same goes for `NUMERIC`.
- Added option `ROW_FORMAT={default | dynamic | fixed | compressed}` to `CREATE_TABLE`.
- `DELETE FROM table_name` didn’t work on temporary tables.
- Changed function `CHAR_LENGTH()` to be multi-byte character safe.
- Added function `ORD(string)`.

D.3.51 Changes in release 3.23.5 (20 Oct 1999)

- Fixed some Y2K problems in the new date handling in 3.23.
- Fixed problem with `SELECT DISTINCT ... ORDER BY RAND()`.
- Added patches by Sergei A. Golubchik for text searching on the `MyISAM` level.
- Fixed cache overflow problem when using full joins without keys.
- Fixed some configure issues.
- Some small changes to make parsing faster.
- Adding a column after the last field with `ALTER TABLE` didn’t work.
- Fixed problem when using an `AUTO_INCREMENT` column in two keys

- With `MyISAM`, you now can have an `AUTO_INCREMENT` column as a key sub part: `CREATE TABLE foo (a INT NOT NULL AUTO_INCREMENT, b CHAR(5), PRIMARY KEY (b,a))`
- Fixed bug in `MyISAM` with packed char keys that could be `NULL`.
- `AS` on field name with `CREATE TABLE table_name SELECT ...` didn't work.
- Allow use of `NATIONAL` and `NCHAR` when defining character columns. This is the same as not using `BINARY`.
- Don't allow `NULL` columns in a `PRIMARY KEY` (only in `UNIQUE` keys).
- Clear `LAST_INSERT_ID()` if one uses this in ODBC: `WHERE auto_increment_column IS NULL`. This seems to fix some problems with Access.
- `SET SQL_AUTO_IS_NULL=0|1` now turns on/off the handling of searching after the last inserted row with `WHERE auto_increment_column IS NULL`.
- Added new variable `concurrency` to `mysqld` for Solaris.
- Added `--relative` option to `mysqladmin` to make `extended-status` more useful to monitor changes.
- Fixed bug when using `COUNT(DISTINCT ...)` on an empty table.
- Added support for the Chinese character set GBK.
- Fixed problem with `LOAD DATA INFILE` and `BLOB` columns.
- Added bit operator `~` (negation).
- Fixed problem with `UDF` functions.

D.3.52 Changes in release 3.23.4 (28 Sep 1999)

- Inserting a `DATETIME` into a `TIME` column no longer will try to store 'days' in it.
- Fixed problem with storage of float/double on little endian machines. (This affected `SUM()`.)
- Added connect timeout on TCP/IP connections.
- Fixed problem with `LIKE "%"` on an index that may have `NULL` values.
- `REVOKE ALL PRIVILEGES` didn't revoke all privileges.
- Allow creation of temporary tables with same name as the original table.
- When granting a user a `GRANT` option for a database, he couldn't grant privileges to other users.
- New command: `SHOW GRANTS FOR user` (by Sinisa).
- New `date_add` syntax: `date/datetime + INTERVAL # interval_type`. By Joshua Chamas.
- Fixed privilege check for `LOAD DATA REPLACE`.
- Automatic fixing of broken include files on Solaris 2.7
- Some configure issues to fix problems with big filesystem detection.
- `REGEXP` is now case-insensitive if you use non-binary strings.

D.3.53 Changes in release 3.23.3

- Added patches for MIT-pthreads on NetBSD.
- Fixed range bug in `MyISAM`.
- `ASC` is now the default again for `ORDER BY`.
- Added `LIMIT` to `UPDATE`.
- Added `mysql_change_user()` function to the MySQL C API.
- Added character set to `SHOW VARIABLES`.
- Added support of `--[whitespace]` comments.
- Allow `INSERT into tbl_name VALUES ()`, that is, you may now specify an empty value list to insert a row in which each column is set to its default value.
- Changed `SUBSTRING(text FROM pos)` to conform to ANSI SQL. (Before this construct returned the rightmost `pos` characters.)
- `SUM()` with `GROUP BY` returned 0 on some systems.
- Changed output for `SHOW TABLE STATUS`.
- Added `DELAY_KEY_WRITE` option to `CREATE TABLE`.
- Allow `AUTO_INCREMENT` on any key part.
- Fixed problem with `YEAR(NOW())` and `YEAR(CURDATE())`.
- Added `CASE` construct.
- New function `COALESCE()`.

D.3.54 Changes in release 3.23.2 (09 Aug 1999)

- Fixed range optimiser bug: `SELECT * FROM table_name WHERE key_part1 >= const AND (key_part2 = const OR key_part2 = const)`. The bug was that some rows could be duplicated in the result.
- Running `myisamchk` without `-a` updated the index distribution incorrectly.
- `SET SQL_LOW_PRIORITY_UPDATES=1` was causing a parse error.
- You can now update index columns that are used in the `WHERE` clause. `UPDATE tbl_name SET KEY=KEY+1 WHERE KEY > 100`
- Date handling should now be a bit faster.
- Added handling of fuzzy dates (dates where day or month is 0), such as `'1999-01-00'`.
- Fixed optimisation of `SELECT ... WHERE key_part1=const1 AND key_part_2=const2 AND key_part1=const4 AND key_part2=const4`; indextype should be `range` instead of `ref`.
- Fixed `egcs` 1.1.2 optimiser bug (when using `BLOB`s) on Linux Alpha.
- Fixed problem with `LOCK TABLES` combined with `DELETE FROM table`.
- `MyISAM` tables now allow keys on `NULL` and `BLOB/TEXT` columns.
- The following join is now much faster: `SELECT ... FROM t1 LEFT JOIN t2 ON ... WHERE t2.not_null_column IS NULL`.
- `ORDER BY` and `GROUP BY` can be done on functions.

- Changed handling of 'const_item' to allow handling of `ORDER BY RAND()`.
- Indexes are now used for `WHERE key_column = function`.
- Indexes are now used for `WHERE key_column = col_name` even if the columns are not identically packed.
- Indexes are now used for `WHERE col_name IS NULL`.
- Changed heap tables to be stored in low_byte_first order (to make it easy to convert to `MyISAM` tables)
- Automatic change of `HEAP` temporary tables to `MyISAM` tables in case of 'table is full' errors.
- Added `--init-file=file_name` option to `mysqld`.
- Added `COUNT(DISTINCT value, [value, ...])`.
- `CREATE TEMPORARY TABLE` now creates a temporary table, in its own namespace, that is automatically deleted if connection is dropped.
- New reserved words (required for `CASE`): `CASE, THEN, WHEN, ELSE and END`.
- New functions `EXPORT_SET()` and `MD5()`.
- Support for the GB2312 Chinese character set.

D.3.55 Changes in release 3.23.1

- Fixed some compilation problems.

D.3.56 Changes in release 3.23.0 (05 Aug 1999: Alpha)

- A new table handler library (`MyISAM`) with a lot of new features. Voir Section 7.1 [MyISAM], page 532.
- You can create in-memory `HEAP` tables which are extremely fast for lookups.
- Support for big files (63-bit) on OSes that support big files.
- New function `LOAD_FILE(filename)` to get the contents of a file as a string value.
- New operator `<=>` which will act as `=` but will return TRUE if both arguments are `NULL`. This is useful for comparing changes between tables.
- Added the ODBC 3.0 `EXTRACT(interval FROM datetime)` function.
- Columns defined as `FLOAT(X)` are not rounded on storage and may be in scientific notation (1.0 E+10) when retrieved.
- `REPLACE` is now faster than before.
- Changed `LIKE` character comparison to behave as `=`; This means that `'e' LIKE 'é'` is now true. (If the line doesn't display correctly, the latter 'e' is a French 'e' with a dot above.)
- `SHOW TABLE STATUS` returns a lot of information about the tables.
- Added `LIKE` to the `SHOW STATUS` command.
- Added `Privileges` column to `SHOW COLUMNS`.
- Added `Packed` and `Comment` columns to `SHOW INDEX`.
- Added comments to tables (with `CREATE TABLE ... COMMENT "xxx"`).

- Added `UNIQUE`, as in `CREATE TABLE table_name (col int not null UNIQUE)`
- New create syntax: `CREATE TABLE table_name SELECT ...`
- New create syntax: `CREATE TABLE IF NOT EXISTS ...`
- Allow creation of `CHAR(0)` columns.
- `DATE_FORMAT()` now requires '`%`' before any format character.
- `DELAYED` is now a reserved word (sorry about that :( ).
- An example procedure is added: `analyse`, file: '`sql_analyse.c`'. This will describe the data in your query. Try the following:

```
SELECT ... FROM ...
WHERE ... PROCEDURE ANALYSE([max elements,[max memory]])
```

  This procedure is extremely useful when you want to check the data in your table!
- `BINARY` cast to force a string to be compared in case-sensitive fashion.
- Added `--skip-show-database` option to `mysqld`.
- Check whether a row has changed in an `UPDATE` now also works with `BLOB/TEXT` columns.
- Added the `INNER` join syntax. **NOTE**: This made `INNER` a reserved word!
- Added support for netmasks to the hostname in the MySQL grant tables. You can specify a netmask using the `IP/NETMASK` syntax.
- If you compare a `NOT NULL DATE/DATETIME` column with `IS NULL`, this is changed to a compare against `0` to satisfy some ODBC applications. (By `shreeve@uci.edu`.)
- `NULL IN (...)` now returns `NULL` instead of `0`. This will ensure that `null_column NOT IN (...)` doesn't match `NULL` values.
- Fix storage of floating-point values in `TIME` columns.
- Changed parsing of `TIME` strings to be more strict. Now the fractional second part is detected (and currently skipped). The following formats are supported:
  - [[DAYS] [H]H:]MM:]SS[.fraction]
  - [[[[H]H]H]H]MM]SS[.fraction]
- Detect (and ignore) fractional second part from `DATETIME`.
- Added the `LOW_PRIORITY` attribute to `LOAD DATA INFILE`.
- The default index name now uses the same case as the column name on which the index name is based.
- Changed default number of connections to 100.
- Use bigger buffers when using `LOAD DATA INFILE`.
- `DECIMAL(x,y)` now works according to ANSI SQL.
- Added aggregate UDF functions. Thanks to Andreas F. Bobak (`bobak@relog.ch`) for this!
- `LAST_INSERT_ID()` is now updated for `INSERT INTO ... SELECT`.
- Some small changes to the join table optimiser to make some joins faster.
- `SELECT DISTINCT` is much faster; it uses the new `UNIQUE` functionality in `MyISAM`. One difference compared to MySQL Version 3.22 is that the output of `DISTINCT` is not sorted anymore.

- All C client API macros are now functions to make shared libraries more reliable. Because of this, you can no longer call `mysql_num_fields()` on a `MYSQL` object, you must use `mysql_field_count()` instead.
- Added use of `LIBWRAP`; patch by Henning P. Schmiedehausen.
- Don't allow `AUTO_INCREMENT` for other than numerical columns.
- Using `AUTO_INCREMENT` will now automatically make the column `NOT NULL`.
- Show `NULL` as the default value for `AUTO_INCREMENT` columns.
- Added `SQL_BIG_RESULT`; `SQL_SMALL_RESULT` is now default.
- Added a shared library RPM. This enhancement was contributed by David Fox (`dsfox@cogsci.ucsd.edu`).
- Added `--enable-large-files` and `--disable-large-files` switches to `configure`. See '`configure.in`' for some systems where this is automatically turned off because of broken implementations.
- Upgraded `readline` to 4.0.
- New `CREATE TABLE` options: `PACK_KEYS` and `CHECKSUM`.
- Added `--default-table-type` option to `mysqld`.

D.4 Changes in release 3.22.x (Older; still supported)

The 3.22 version has faster and safer connect code than version 3.21, as well as a lot of new nice enhancements. As there aren't really any major changes, upgrading from 3.21 to 3.22 should be very easy and painless. Voir Section 2.5.3 [Upgrading-from-3.21], page 111.

D.4.1 Changes in release 3.22.35

- Fixed problem with `STD()`.
- Merged changes from the newest `ISAM` library from 3.23.
- Fixed problem with `INSERT DELAYED`.
- Fixed a bug core dump when using a `LEFT JOIN/STRAIGHT_JOIN` on a table with only one row.

D.4.2 Changes in release 3.22.34

- Fixed problem with `GROUP BY` on `TINYBLOB` columns; this caused bugzilla to not show rows in some queries.
- Had to do total recompile of the Windows binary version as VC++ didn't compile all relevant files for 3.22.33 :(

D.4.3 Changes in release 3.22.33

- Fixed problems in Windows when locking tables with `LOCK TABLE`.
- Quicker kill of `SELECT DISTINCT` queries.

D.4.4 Changes in release 3.22.32 (14 Feb 2000)

- Fixed problem when storing numbers in timestamps.
- Fix problem with timezones that have half hour offsets.
- Added `mysqlhotcopy`, a fast online hot-backup utility for local MySQL databases. By Tim Bunce.
- New more secure `mysqlaccess`. Thanks to Steve Harvey for this.
- Fixed security problem in the protocol regarding password checking.
- Fixed problem that affected queries that did arithmetic on `GROUP` functions.
- Fixed a bug in the `ISAM` code when deleting rows on tables with packed indexes.

D.4.5 Changes in release 3.22.31

- A few small fixes for the Windows version.

D.4.6 Changes in release 3.22.30

- Fixed optimiser problem on `SELECT` when using many overlapping indexes.
- Disabled floating-point exceptions for FreeBSD to fix core dump when doing `SELECT FLOOR(POW(2,63))`.
- Added print of default arguments options to all clients.
- Fixed critical problem with the `WITH GRANT OPTION` option.
- Fixed non-critical Y2K problem when writing short date to log files.

D.4.7 Changes in release 3.22.29 (02 Jan 2000)

- Upgraded the configure and include files to match the latest 3.23 version. This should increase portability and make it easier to build shared libraries.
- Added latest patches to MIT-pthreads for NetBSD.
- Fixed problem with timezones that are < GMT -11.
- Fixed a bug when deleting packed keys in NISAM.
- Fixed problem that could cause MySQL to touch freed memory when doing very complicated `GROUP BY` queries.
- Fixed core dump if you got a crashed table where an `ENUM` field value was too big.
- Added `mysqlshutdown.exe` and `mysqlwatch.exe` to the Windows distribution.
- Fixed problem when doing `ORDER BY` on a reference key.
- Fixed that `INSERT DELAYED` doesn't update timestamps that are given.

D.4.8 Changes in release 3.22.28 (20 Oct 1999)

- Fixed problem with `LEFT JOIN` and `COUNT()` on a column which was declared `NULL +` and it had a `DEFAULT` value.
- Fixed core dump problem when using `CONCAT()` in a `WHERE` clause.
- Fixed problem with `AVG()` and `STD()` with `NULL` values.

D.4.9 Changes in release 3.22.27

- Fixed prototype in '`my_ctype.h`' when using other character sets.
- Some configure issues to fix problems with big filesystem detection.
- Fixed problem when sorting on big `BLOB` columns.
- `ROUND()` will now work on Windows.

D.4.10 Changes in release 3.22.26 (16 Sep 1999)

- Fixed core dump with empty `BLOB/TEXT` column argument to `REVERSE()`.
- Extended `/*! */` with version numbers.
- Changed `SUBSTRING(text FROM pos)` to conform to ANSI SQL. (Before this construct returned the rightmost 'pos' characters.)
- Fixed problem with `LOCK TABLES` combined with `DELETE FROM table`
- Fixed problem that `INSERT ... SELECT` didn't use `BIG_TABLES`.
- `SET SQL_LOW_PRIORITY_UPDATES=#` didn't work.
- Password wasn't updated correctly if privileges didn't change on: `GRANT ... IDENTIFIED BY`
- Fixed range optimiser bug in `SELECT * FROM table_name WHERE key_part1 >= const AND (key_part2 = const OR key_part2 = const)`.
- Fixed bug in compression key handling in `ISAM`.

D.4.11 Changes in release 3.22.25

- Fixed some small problems with the installation.

D.4.12 Changes in release 3.22.24 (05 Jul 1999)

- `DATA` is not a reserved word anymore.
- Fixed optimiser bug with tables with only one row.
- Fixed bug when using `LOCK TABLES table_name READ; FLUSH TABLES;`
- Applied some patches for HP-UX.
- `isamchk` should now work on Windows.
- Changed '`configure`' to not use big file handling on Linux as this crashes some RedHat 6.0 systems

D.4.13 Changes in release 3.22.23 (08 Jun 1999)

- Upgraded to use Autoconf 2.13, Automake 1.4 and `libtool` 1.3.2.
- Better support for SCO in `configure`.
- Added option `--defaults-file=###` to option file handling to force use of only one specific option file.
- Extended `CREATE` syntax to ignore MySQL Version 3.23 keywords.

- Fixed deadlock problem when using `INSERT DELAYED` on a table locked with `LOCK TABLES`.
- Fixed deadlock problem when using `DROP TABLE` on a table that was locked by another thread.
- Add logging of `GRANT/REVOKE` commands in the update log.
- Fixed `isamchk` to detect a new error condition.
- Fixed bug in `NATURAL LEFT JOIN`.

D.4.14 Changes in release 3.22.22 (30 Apr 1999)

- Fixed problem in the C API when you called `mysql_close()` directly after `mysql_init()`.
- Better client error message when you can't open socket.
- Fixed `delayed_insert_thread` counting when you couldn't create a new delayed_insert thread.
- Fixed bug in `CONCAT()` with many arguments.
- Added patches for DEC 3.2 and SCO.
- Fixed path-bug when installing MySQL as a service on NT.
- The MySQL-Windows version is now compiled with VC++ 6.0 instead of with VC++ 5.0.
- New installation setup for MySQL-Windows.

D.4.15 Changes in release 3.22.21

- Fixed problem with `DELETE FROM TABLE` when table was locked by another thread.
- Fixed bug in `LEFT JOIN` involving empty tables.
- Changed the `mysql.db` column from `CHAR(32)` to `CHAR(60)`.
- `MODIFY` and `DELAYED` are not reserved words anymore.
- Fixed a bug when storing days in a `TIME` column.
- Fixed a problem with `Host '...' is not allowed to connect to this MySQL server` after one had inserted a new MySQL user with a `GRANT` command.
- Changed to use `TCP_NODELAY` also on Linux (should give faster TCP/IP connections).

D.4.16 Changes in release 3.22.20 (18 Mar 1999)

- Fixed `STD()` for big tables when result should be 0.
- The update log didn't have newlines on some operating systems.
- `INSERT DELAYED` had some garbage at end in the update log.

D.4.17 Changes in release 3.22.19 (Mar 1999: Stable)

- Fixed bug in `mysql_install_db` (from 3.22.17).
- Changed default key cache size to 8M.
- Fixed problem with queries that needed temporary tables with `BLOB` columns.

D.4.18 Changes in release 3.22.18

- Fixes a fatal problem in 3.22.17 on Linux; after `shutdown` not all threads died properly.
- Added option `-O flush_time=#` to `mysqld`. This is mostly useful on Windows and tells how often MySQL should close all unused tables and flush all updated tables to disk.
- Fixed problem that a `VARCHAR` column compared with `CHAR` column didn't use keys efficiently.

D.4.19 Changes in release 3.22.17

- Fixed a core dump problem when using `--log-update` and connecting without a default database.
- Fixed some `configure` and portability problems.
- Using `LEFT JOIN` on tables that had circular dependencies caused `mysqld` to hang forever.

D.4.20 Changes in release 3.22.16 (Feb 1999: Gamma)

- `mysqladmin processlist` could kill the server if a new user logged in.
- `DELETE FROM tbl_name WHERE key_column=col_name` didn't find any matching rows. Fixed.
- `DATE_ADD(column, ...)` didn't work.
- `INSERT DELAYED` could deadlock with status 'upgrading lock'
- Extended `ENCRYPT()` to take longer salt strings than 2 characters.
- `longlong2str` is now much faster than before. For `Intel x86` platforms, this function is written in optimised assembler.
- Added the `MODIFY` keyword to `ALTER TABLE`.

D.4.21 Changes in release 3.22.15

- `GRANT` used with `IDENTIFIED BY` didn't take effect until privileges were flushed.
- Name change of some variables in `SHOW STATUS`.
- Fixed problem with `ORDER BY` with 'only index' optimisation when there were multiple key definitions for a used column.
- `DATE` and `DATETIME` columns are now up to 5 times faster than before.
- `INSERT DELAYED` can be used to let the client do other things while the server inserts rows into a table.
- `LEFT JOIN USING (col1,col2)` didn't work if one used it with tables from 2 different databases.
- `LOAD DATA LOCAL INFILE` didn't work in the Unix version because of a missing file.
- Fixed problems with `VARCHAR/BLOB` on very short rows (< 4 bytes); error 127 could occur when deleting rows.
- Updating `BLOB/TEXT` through formulas didn't work for short (< 256 char) strings.

- When you did a `GRANT` on a new host, `mysqld` could die on the first connect from this host.
- Fixed bug when one used `ORDER BY` on column name that was the same name as an alias.
- Added `BENCHMARK(loop_count,expression)` function to time expressions.

D.4.22 Changes in release 3.22.14

- Allow empty arguments to `mysqld` to make it easier to start from shell scripts.
- Setting a `TIMESTAMP` column to `NULL` didn't record the timestamp value in the update log.
- Fixed lock handler bug when one did `INSERT INTO TABLE ... SELECT ... GROUP BY`.
- Added a patch for `localtime_r()` on Windows so that it will not crash anymore if your date is > 2039, but instead will return a time of all zero.
- Names for user-defined functions are no longer case-sensitive.
- Added escape of `^Z` (ASCII 26) to `\Z` as `^Z` doesn't work with pipes on Windows.
- `mysql_fix_privileges` adds a new column to the `mysql.func` to support aggregate UDF functions in future MySQL releases.

D.4.23 Changes in release 3.22.13

- Saving `NOW()`, `CURDATE()` or `CURTIME()` directly in a column didn't work.
- `SELECT COUNT(*) ... LEFT JOIN ...` didn't work with no `WHERE` part.
- Updated '`config.guess`' to allow MySQL to configure on UnixWare 7.0.x.
- Changed the implementation of `pthread_cond()` on the Windows version. `get_lock()` now correctly times out on Windows!

D.4.24 Changes in release 3.22.12

- Fixed problem when using `DATE_ADD()` and `DATE_SUB()` in a `WHERE` clause.
- You can now set the password for a user with the `GRANT ... TO user IDENTIFIED BY 'password'` syntax.
- Fixed bug in `GRANT` checking with `SELECT` on many tables.
- Added missing file `mysql_fix_privilege_tables` to the RPM distribution. This is not run by default because it relies on the client package.
- Added option `SQL_SMALL_RESULT` to `SELECT` to force use of fast temporary tables when you know that the result set will be small.
- Allow use of negative real numbers without a decimal point.
- Day number is now adjusted to maximum days in month if the resulting month after `DATE_ADD`/`DATE_SUB()` doesn't have enough days.
- Fix that `GRANT` compares columns in case-insensitive fashion.
- Fixed a bug in '`sql_list.h`' that made `ALTER TABLE` dump core in some contexts.

- The hostname in `user@hostname` can now include '`.`' and '`-`' without quotes in the context of the `GRANT`, `REVOKE` and `SET PASSWORD FOR ...` statements.
- Fix for `isamchk` for tables which need big temporary files.

D.4.25 Changes in release 3.22.11

- **Important**: You must run the `mysql_fix_privilege_tables` script when you upgrade to this version! This is needed because of the new `GRANT` system. If you don't do this, you will get `Access denied` when you try to use `ALTER TABLE`, `CREATE INDEX`, or `DROP INDEX`.
- `GRANT` to allow/deny users table and column access.
- Changed `USER()` to return a value in `user@host` format. Formerly it returned only `user`.
- Changed the syntax for how to set `PASSWORD` for another user.
- New command `FLUSH STATUS` that resets most status variables to zero.
- New status variables: `aborted_threads`, `aborted_connects`.
- New option variable: `connection_timeout`.
- Added support for Thai sorting (by Pruet Boonma `pruet@ds90.intanon.nectec.or.th`).
- Slovak and japanese error messages.
- Configuration and portability fixes.
- Added option `SET SQL_WARNINGS=1` to get a warning count also for simple inserts.
- MySQL now uses `SIGTERM` instead of `SIGQUIT` with shutdown to work better on FreeBSD.
- Added option `\G` (print vertically) to `mysql`.
- `SELECT HIGH_PRIORITY ...` killed `mysqld`.
- `IS NULL` on a `AUTO_INCREMENT` column in a `LEFT JOIN` didn't work as expected.
- New function `MAKE_SET()`.

D.4.26 Changes in release 3.22.10

- `mysql_install_db` no longer starts the MySQL server! You should start `mysqld` with `safe_mysqld` after installing it! The MySQL RPM will, however, start the server as before.
- Added `--bootstrap` option to `mysqld` and recoded `mysql_install_db` to use it. This will make it easier to install MySQL with RPMs.
- Changed `+`, `-` (sign and minus), `*`, `/`, `%`, `ABS()` and `MOD()` to be `BIGINT` aware (64-bit safe).
- Fixed a bug in `ALTER TABLE` that caused `mysqld` to crash.
- MySQL now always reports the conflicting key values when a duplicate key entry occurs. (Before this was only reported for `INSERT`.)
- New syntax: `INSERT INTO tbl_name SET col_name=value, col_name=value, ...`
- Most errors in the '`.err`' log are now prefixed with a time stamp.

- Added option `MYSQL_INIT_COMMAND` to `mysql_options()` to make a query on connect or reconnect.
- Added option `MYSQL_READ_DEFAULT_FILE` and `MYSQL_READ_DEFAULT_GROUP` to `mysql_options()` to read the following parameters from the MySQL option files: `port`, `socket`, `compress`, `password`, `pipe`, `timeout`, `user`, `init-command`, `host` and `database`.
- Added `maybe_null` to the UDF structure.
- Added option `IGNORE` to `INSERT` statements with many rows.
- Fixed some problems with sorting of the `koi8` character sets; users of `koi8` **must** run `isamchk -rq` on each table that has an index on a `CHAR` or `VARCHAR` column.
- New script `mysql_setpermission`, by Luuk de Boer. It allows easy creation of new users with permissions for specific databases.
- Allow use of hexadecimal strings (0x...) when specifying a constant string (like in the column separators with `LOAD DATA INFILE`).
- Ported to OS/2 (thanks to Antony T. Curtis `antony.curtis@olcs.net`).
- Added more variables to `SHOW STATUS` and changed format of output to be like `SHOW VARIABLES`.
- Added `extended-status` command to `mysqladmin` which will show the new status variables.

D.4.27 Changes in release 3.22.9

- `SET SQL_LOG_UPDATE=0` caused a lockup of the server.
- New SQL command: `FLUSH [ TABLES | HOSTS | LOGS | PRIVILEGES ] [, ...]`
- New SQL command: `KILL thread_id`.
- Added casts and changed include files to make MySQL easier to compile on AIX and DEC OSF/1 4.x
- Fixed conversion problem when using `ALTER TABLE` from a `INT` to a short `CHAR()` column.
- Added `SELECT HIGH_PRIORITY`; this will get a lock for the `SELECT` even if there is a thread waiting for another `SELECT` to get a `WRITE LOCK`.
- Moved `wild_compare()` to string class to be able to use `LIKE` on `BLOB/TEXT` columns with `\0`.
- Added `ESCAPE` option to `LIKE`.
- Added a lot more output to `mysqladmin debug`.
- You can now start `mysqld` on Windows with the `--flush` option. This will flush all tables to disk after each update. This makes things much safer on the Windows platforms but also **much** slower.

D.4.28 Changes in release 3.22.8

- Czech character sets should now work much better. You must also install `http://www.mysql.com/Downloads/Patches/czech-3.22.8-patch`. This patch

should also be installed if you are using a character set which uses `my_strcoll()`! The patch should always be safe to install (for any system), but as this patch changes `ISAM` internals it's not yet in the default distribution.

- `DATE_ADD()` and `DATE_SUB()` didn't work with group functions.
- `mysql` will now also try to reconnect on `USE DATABASE` commands.
- Fix problem with `ORDER BY` and `LEFT JOIN` and `const` tables.
- Fixed problem with `ORDER BY` if the first `ORDER BY` column was a key and the rest of the `ORDER BY` columns wasn't part of the key.
- Fixed a big problem with `OPTIMIZE TABLE`.
- MySQL clients on NT will now by default first try to connect with named pipes and after this with TCP/IP.
- Fixed a problem with `DROP TABLE` and `mysqladmin shutdown` on Windows (a fatal bug from 3.22.6).
- Fixed problems with `TIME columns` and negative strings.
- Added an extra thread signal loop on shutdown to avoid some error messages from the client.
- MySQL now uses the next available number as extension for the update log file.
- Added patches for UNIXWARE 7.

D.4.29 Changes in release 3.22.7 (Sep 1998: Beta)

- Added `LIMIT` clause for the `DELETE` statement.
- You can now use the `/*! ... */` syntax to hide MySQL-specific keywords when you write portable code. MySQL will parse the code inside the comments as if the surrounding `/*!` and `*/` comment characters didn't exist.
- `OPTIMIZE TABLE tbl_name` can now be used to reclaim disk space after many deletes. Currently, this uses `ALTER TABLE` to regenerate the table, but in the future it will use an integrated `isamchk` for more speed.
- Upgraded `libtool` to get the configure more portable.
- Fixed slow `UPDATE` and `DELETE` operations when using `DATETIME` or `DATE` keys.
- Changed optimiser to make it better at deciding when to do a full join and when using keys.
- You can now use `mysqladmin proc` to display information about your own threads. Only users with the `PROCESS` privilege can get information about all threads. (In 4.0.2 one needs the `SUPER` privilege for this.)
- Added handling of formats `YYMMDD`, `YYYYMMDD`, `YYMMDDHHMMSS` for numbers when using `DATETIME` and `TIMESTAMP` types. (Formerly these formats only worked with strings.)
- Added connect option `CLIENT_IGNORE_SPACE` to allow use of spaces after function names and before '`(`' (Powerbuilder requires this). This will make all function names reserved words.
- Added the `--log-long-format` option to `mysqld` to enable timestamps and INSERT_ID's in the update log.

- Added `--where` option to `mysqldump` (patch by Jim Faucette).
- The lexical analyser now uses "perfect hashing" for faster parsing of SQL statements.

D.4.30 Changes in release 3.22.6

- Faster `mysqldump`.
- For the `LOAD DATA INFILE` statement, you can now use the new `LOCAL` keyword to read the file from the client. `mysqlimport` will automatically use `LOCAL` when importing with the TCP/IP protocol.
- Fixed small optimise problem when updating keys.
- Changed makefiles to support shared libraries.
- MySQL-NT can now use named pipes, which means that you can now use MySQL-NT without having to install TCP/IP.

D.4.31 Changes in release 3.22.5

- All table lock handing is changed to avoid some very subtle deadlocks when using `DROP TABLE`, `ALTER TABLE`, `DELETE FROM TABLE` and `mysqladmin flush-tables` under heavy usage. Changed locking code to get better handling of locks of different types.
- Updated `DBI` to 1.00 and `DBD` to 1.2.0.
- Added a check that the error message file contains error messages suitable for the current version of `mysqld`. (To avoid errors if you accidentally try to use an old error message file.)
- All count structures in the client (`affected_rows()`, `insert_id()`, ...) are now of type `BIGINT` to allow 64-bit values to be used. This required a minor change in the MySQL protocol which should affect only old clients when using tables with `AUTO_INCREMENT` values > 16M.
- The return type of `mysql_fetch_lengths()` has changed from `uint *` to `ulong *`. This may give a warning for old clients but should work on most machines.
- Change `mysys` and `dbug` libraries to allocate all thread variables in one struct. This makes it easier to make a threaded '`libmysql.dll`' library.
- Use the result from `gethostname()` (instead of `uname()`) when constructing '`.pid`' file names.
- New better compressed server/client protocol.
- `COUNT()`, `STD()` and `AVG()` are extended to handle more than 4G rows.
- You can now store values in the range `-838:59:59 <= x <= 838:59:59` in a `TIME` column.
- **Warning: incompatible change!!** If you set a `TIME` column to too short a value, MySQL now assumes the value is given as: `[[[D ]HH:]MM:]SS` instead of `HH[:MM[:SS]]`.
- `TIME_TO_SEC()` and `SEC_TO_TIME()` can now handle negative times and hours up to 32767.
- Added new option `SET SQL_LOG_UPDATE={0|1}` to allow users with the `PROCESS` privilege to bypass the update log. (Modified patch from Sergey A Mukhin `violet@rosnet.net`.)

- Fixed fatal bug in `LPAD()`.
- Initialise line buffer in '`mysql.cc`' to make `BLOB` reading from pipes safer.
- Added `-O max_connect_errors=#` option to `mysqld`. Connect errors are now reset for each correct connection.
- Increased the default value of `max_allowed_packet` to `1M` in `mysqld`.
- Added `--low-priority-updates` option to `mysqld`, to give table-modifying operations (`INSERT`, `REPLACE`, `UPDATE`, `DELETE`) lower priority than retrievals. You can now use `{INSERT | REPLACE | UPDATE | DELETE} LOW_PRIORITY ...` You can also use `SET SQL_LOW_PRIORITY_UPDATES={0|1}` to change the priority for one thread. One side effect is that `LOW_PRIORITY` is now a reserved word. :(
- Add support for `INSERT INTO table ... VALUES(...),(...),(...)`, to allow inserting multiple rows with a single statement.
- `INSERT INTO tbl_name` is now also cached when used with `LOCK TABLES`. (Previously only `INSERT ... SELECT` and `LOAD DATA INFILE` were cached.)
- Allow `GROUP BY` functions with `HAVING`:

```
mysql> SELECT col FROM table GROUP BY col HAVING COUNT(*)>0;
```

- `mysqld` will now ignore trailing '`;`' characters in queries. This is to make it easier to migrate from some other SQL servers that require the trailing '`;`'.
- Fix for corrupted fixed-format output generated by `SELECT INTO OUTFILE`.
- **Warning: incompatible change!** Added Oracle `GREATEST()` and `LEAST()` functions. You must now use these instead of the `MAX()` and `MIN()` functions to get the largest/smallest value from a list of values. These can now handle `REAL`, `BIGINT` and string (`CHAR` or `VARCHAR`) values.
- **Warning: incompatible change!** `DAYOFWEEK()` had offset 0 for Sunday. Changed the offset to 1.
- Give an error for queries that mix `GROUP BY` columns and fields when there is no `GROUP BY` specification.
- Added `--vertical` option to `mysql`, for printing results in vertical mode.
- Index-only optimisation; some queries are now resolved using only indexes. Until MySQL 4.0, this works only for numeric columns. Voir Section 5.4.3 [MySQL indexes], page 384.
- Lots of new benchmarks.
- A new C API chapter and lots of other improvements in the manual.

D.4.32 Changes in release 3.22.4

- Added `--tmpdir` option to `mysqld`, for specifying the location of the temporary file directory.
- MySQL now automatically changes a query from an ODBC client:

```
SELECT ... FROM table WHERE auto_increment_column IS NULL
```

  to:

```
SELECT ... FROM table WHERE auto_increment_column == LAST_INSERT_ID()
```

This allows some ODBC programs (Delphi, Access) to retrieve the newly inserted row to fetch the `AUTO_INCREMENT` id.

- `DROP TABLE` now waits for all users to free a table before deleting it.
- Fixed small memory leak in the new connect protocol.
- New functions `BIN()`, `OCT()`, `HEX()` and `CONV()` for converting between different number bases.
- Added function `SUBSTRING()` with 2 arguments.
- If you created a table with a record length smaller than 5, you couldn't delete rows from the table.
- Added optimisation to remove const reference tables from `ORDER BY` and `GROUP BY`.
- `mysqld` now automatically disables system locking on Linux and Windows, and for systems that use MIT-pthreads. You can force the use of locking with the `--enable-external-locking` option.
- Added `--console` option to `mysqld`, to force a console window (for error messages) when using Windows.
- Fixed table locks for Windows.
- Allow '`$`' in identifiers.
- Changed name of user-specific configuration file from '`my.cnf`' to '`.my.cnf`' (Unix only).
- Added `DATE_ADD()` and `DATE_SUB()` functions.

D.4.33 Changes in release 3.22.3

- Fixed a lock problem (bug in MySQL Version 3.22.1) when closing temporary tables.
- Added missing `mysql_ping()` to the client library.
- Added `--compress` option to all MySQL clients.
- Changed `byte` to `char` in '`mysql.h`' and '`mysql_com.h`'.

D.4.34 Changes in release 3.22.2

- Searching on multiple constant keys that matched more than 30% of the rows didn't always use the best possible key.
- New functions `<<`, `>>`, `RPAD()` and `LPAD()`.
- You can now save default options (like passwords) in a configuration file ('`my.cnf`').
- Lots of small changes to get `ORDER BY` to work when no records are found when using fields that are not in `GROUP BY` (MySQL extension).
- Added `--chroot` option to `mysqld`, to start `mysqld` in a chroot environment (by Nikki Chumakov `nikkic@cityline.ru`).
- Trailing spaces are now ignored when comparing case-sensitive strings; this should fix some problems with ODBC and flag 512!
- Fixed a core dump bug in the range optimiser.

- Added `--one-thread` option to `mysqld`, for debugging with LinuxThreads (or `glibc`). (This replaces the `-T32` flag)
- Added `DROP TABLE IF EXISTS` to prevent an error from occurring if the table doesn't exist.
- `IF` and `EXISTS` are now reserved words (they would have to be sooner or later).
- Added lots of new options to `mysqldump`.
- Server error messages are now in '`mysqld_error.h`'.
- The server/client protocol now supports compression.
- All bug fixes from MySQL Version 3.21.32.

D.4.35 Changes in release 3.22.1 (Jun 1998: Alpha)

- Added new C API function `mysql_ping()`.
- Added new API functions `mysql_init()` and `mysql_options()`. You now MUST call `mysql_init()` before you call `mysql_real_connect()`. You don't have to call `mysql_init()` if you only use `mysql_connect()`.
- Added `mysql_options(...,MYSQL_OPT_CONNECT_TIMEOUT,...)` so you can set a timeout for connecting to a server.
- Added `--timeout` option to `mysqladmin`, as a test of `mysql_options()`.
- Added `AFTER column` and `FIRST` options to `ALTER TABLE ... ADD columns`. This makes it possible to add a new column at some specific location within a row in an existing table.
- `WEEK()` now takes an optional argument to allow handling of weeks when the week starts on Monday (some European countries). By default, `WEEK()` assumes the week starts on Sunday.
- `TIME` columns weren't stored properly (bug in MySQL Version 3.22.0).
- `UPDATE` now returns information about how many rows were matched and updated, and how many "warnings" occurred when doing the update.
- Fixed incorrect result from `FORMAT(-100,2)`.
- `ENUM` and `SET` columns were compared in binary (case-sensitive) fashion; changed to be case-insensitive.

D.4.36 Changes in release 3.22.0

- New (backward-compatible) connect protocol that allows you to specify the database to use when connecting, to get much faster connections to a specific database.

  The `mysql_real_connect()` call is changed to:

  ```
  mysql_real_connect(MYSQL *mysql, const char *host, const char *user,
                     const char *passwd, const char *db, uint port,
                     const char *unix_socket, uint client_flag)
  ```

- Each connection is handled by its own thread, rather than by the master `accept()` thread. This fixes permanently the telnet bug that was a topic on the mail list some time ago.

- All TCP/IP connections are now checked with backward-resolution of the hostname to get better security. `mysqld` now has a local hostname resolver cache so connections should actually be faster than before, even with this feature.
- A site automatically will be blocked from future connections if someone repeatedly connects with an "improper header" (like when one uses telnet).
- You can now refer to tables in different databases with references of the form `tbl_name@db_name` or `db_name.tbl_name`. This makes it possible to give a user read access to some tables and write access to others simply by keeping them in different databases!
- Added `--user` option to `mysqld`, to allow it to run as another Unix user (if it is started as the Unix `root` user).
- Added caching of users and access rights (for faster access rights checking)
- Normal users (not anonymous ones) can change their password with `mysqladmin password 'new_password'`. This uses encrypted passwords that are not logged in the normal MySQL log!
- All important string functions are now coded in assembler for x86 Linux machines. This gives a speedup of 10% in many cases.
- For tables that have many columns, the column names are now hashed for much faster column name lookup (this will speed up some benchmark tests a lot!)
- Some benchmarks are changed to get better individual timing. (Some loops were so short that a specific test took < 2 seconds. The loops have been changed to take about 20 seconds to make it easier to compare different databases. A test that took 1-2 seconds before now takes 11-24 seconds, which is much better)
- Re-arranged `SELECT` code to handle some very specific queries involving group functions (like `COUNT(*)`) without a `GROUP BY` but with `HAVING`. The following now works:

```
mysql> SELECT COUNT(*) as C FROM table HAVING C > 1;
```

- Changed the protocol for field functions to be faster and avoid some calls to `malloc()`.
- Added `-T32` option to `mysqld`, for running all queries under the main thread. This makes it possible to debug `mysqld` under Linux with `gdb`!
- Added optimisation of `not_null_column IS NULL` (needed for some Access queries).
- Allow `STRAIGHT_JOIN` to be used between two tables to force the optimiser to join them in a specific order.
- String functions now return `VARCHAR` rather than `CHAR` and the column type is now `VARCHAR` for fields saved as `VARCHAR`. This should make the `MyODBC` driver better, but may break some old MySQL clients that don't handle `FIELD_TYPE_VARCHAR` the same way as `FIELD_TYPE_CHAR`.
- `CREATE INDEX` and `DROP INDEX` are now implemented through `ALTER TABLE`. `CREATE TABLE` is still the recommended (fast) way to create indexes.
- Added `--set-variable` option `wait_timeout` to `mysqld`.
- Added time column to `mysqladmin processlist` to show how long a query has taken or how long a thread has slept.
- Added lots of new variables to `show variables` and some new to `show status`.
- Added new type `YEAR`. `YEAR` is stored in 1 byte with allowable values of 0, and 1901 to 2155.

- Added new `DATE` type that is stored in 3 bytes rather than 4 bytes. All new tables are created with the new date type if you don't use the `--old-protocol` option to `mysqld`.
- Fixed bug in record caches; for some queries, you could get `Error from table handler: #` on some operating systems.
- Added `--enable-assembler` option to `configure`, for x86 machines (tested on Linux + `gcc`). This will enable assembler functions for the most important string functions for more speed!

D.5 Changes in release 3.21.x

Version 3.21 is quite old now, and should be avoided if possible. This information is kept here for historical purposes only.

D.5.1 Changes in release 3.21.33

- Fixed problem when sending `SIGHUP` to `mysqld`; `mysqld` core dumped when starting from boot on some systems.
- Fixed problem with losing a little memory for some connections.
- `DELETE FROM tbl_name` without a `WHERE` condition is now done the long way when you use `LOCK TABLES` or if the table is in use, to avoid race conditions.
- `INSERT INTO TABLE (timestamp_column) VALUES (NULL);` didn't set timestamp.

D.5.2 Changes in release 3.21.32

- Fixed some possible race conditions when doing many reopen/close on the same tables under heavy load! This can happen if you execute `mysqladmin refresh` often. This could in some very rare cases corrupt the header of the index file and cause error 126 or 138.
- Fixed fatal bug in `refresh()` when running with the `--skip-external-locking` option. There was a "very small" time gap after a `mysqladmin refresh` when a table could be corrupted if one thread updated a table while another thread did `mysqladmin refresh` and another thread started a new update ont the same table before the first thread had finished. A refresh (or `--flush-tables`) will now not return until all used tables are closed!
- `SELECT DISTINCT` with a `WHERE` clause that didn't match any rows returned a row in some contexts (bug only in 3.21.31).
- `GROUP BY + ORDER BY` returned one empty row when no rows where found.
- Fixed a bug in the range optimiser that wrote `Use_count: Wrong count for ...` in the error log file.

D.5.3 Changes in release 3.21.31

- Fixed a sign extension problem for the `TINYINT` type on Irix.
- Fixed problem with `LEFT("constant_string",function)`.

- Fixed problem with `FIND_IN_SET()`.
- `LEFT JOIN` core dumped if the second table is used with a constant `WHERE/ON` expression that uniquely identifies one record.
- Fixed problems with `DATE_FORMAT()` and incorrect dates. `DATE_FORMAT()` now ignores `'%'` to make it possible to extend it more easily in the future.

D.5.4 Changes in release 3.21.30

- `mysql` now returns an exit code > 0 if the query returned an error.
- Saving of command-line history to file in `mysql` client. By Tommy Larsen `tommy@mix.hive.no`.
- Fixed problem with empty lines that were ignored in '`mysql.cc`'.
- Save the pid of the signal handler thread in the pid file instead of the pid of the main thread.
- Added patch by `tommy@valley.ne.jp` to support Japanese characters SJIS and UJIS.
- Changed `safe_mysqld` to redirect startup messages to `'hostname'.err` instead of `'hostname'.log` to reclaim file space on `mysqladmin refresh`.
- `ENUM` always had the first entry as default value.
- `ALTER TABLE` wrote two entries to the update log.
- `sql_acc()` now closes the `mysql` grant tables after a reload to save table space and memory.
- Changed `LOAD DATA` to use less memory with tables and `BLOB` columns.
- Sorting on a function which made a division / 0 produced a wrong set in some cases.
- Fixed `SELECT` problem with `LEFT()` when using the `czech` character set.
- Fixed problem in `isamchk`; it couldn't repair a packed table in a very unusual case.
- `SELECT` statements with `&` or `|` (bit functions) failed on columns with `NULL` values.
- When comparing a field = field, where one of the fields was a part key, only the length of the part key was compared.

D.5.5 Changes in release 3.21.29

- `LOCK TABLES + DELETE from tbl_name` never removed locks properly.
- Fixed problem when grouping on an `OR` function.
- Fixed permission problem with `umask()` and creating new databases.
- Fixed permission problem on result file with `SELECT ... INTO OUTFILE ...`
- Fixed problem in range optimiser (core dump) for a very complex query.
- Fixed problem when using `MIN(integer)` or `MAX(integer)` in `GROUP BY`.
- Fixed bug on Alpha when using integer keys. (Other keys worked on Alpha.)
- Fixed bug in `WEEK("XXXX-xx-01")`.

D.5.6 Changes in release 3.21.28

- Fixed socket permission (clients couldn't connect to Unix socket on Linux).
- Fixed bug in record caches; for some queries, you could get `Error from table handler: #` on some operating systems.

D.5.7 Changes in release 3.21.27

- Added user level lock functions `GET_LOCK(string,timeout)`, `RELEASE_LOCK(string)`.
- Added `Opened_tables` to `show status`.
- Changed connect timeout to 3 seconds to make it somewhat harder for crackers to kill `mysqld` through telnet + TCP/IP.
- Fixed bug in range optimiser when using `WHERE key_part_1 >= something AND key_part_2 <= something_else`.
- Changed `configure` for detection of FreeBSD 3.0 9803xx and above
- `WHERE` with `string_col_key = constant_string` didn't always find all rows if the column had many values differing only with characters of the same sort value (like e and é).
- Strings keys looked up with 'ref' were not compared in case-sensitive fashion.
- Added `umask()` to make log files non-readable for normal users.
- Ignore users with old (8-byte) password on startup if not using `--old-protocol` option to `mysqld`.
- `SELECT` which matched all key fields returned the values in the case of the matched values, not of the found values. (Minor problem.)

D.5.8 Changes in release 3.21.26

- `FROM_DAYS(0)` now returns "0000-00-00".
- In `DATE_FORMAT()`, PM and AM were swapped for hours 00 and 12.
- Extended the default maximum key size to 256.
- Fixed bug when using `BLOB/TEXT` in `GROUP BY` with many tables.
- An `ENUM` field that is not declared `NOT NULL` has `NULL` as the default value. (Previously, the default value was the first enumeration value.)
- Fixed bug in the join optimiser code when using many part keys on the same key: `INDEX (Organisation,Surname(35),Initials(35))`.
- Added some tests to the table order optimiser to get some cases with `SELECT ... FROM many_tables` much faster.
- Added a retry loop around `accept()` to possibly fix some problems on some Linux machines.

D.5.9 Changes in release 3.21.25

- Changed `typedef 'string'` to `typedef 'my_string'` for better portability.

- You can now kill threads that are waiting on a disk-full condition.
- Fixed some problems with UDF functions.
- Added long options to `isamchk`. Try `isamchk --help`.
- Fixed a bug when using 8 bytes long (alpha); `filesort()` didn't work. Affects `DISTINCT`, `ORDER BY` and `GROUP BY` on 64-bit processors.

D.5.10 Changes in release 3.21.24

- Dynamic loadable functions. Based on source from Alexis Mikhailov.
- You couldn't delete from a table if no one had done a `SELECT` on the table.
- Fixed problem with range optimiser with many `OR` operators on key parts inside each other.
- Recoded `MIN()` and `MAX()` to work properly with strings and `HAVING`.
- Changed default umask value for new files from `0664` to `0660`.
- Fixed problem with `LEFT JOIN` and constant expressions in the `ON` part.
- Added Italian error messages from `brenno@dewinter.com`.
- `configure` now works better on OSF/1 (tested on 4.0D).
- Added hooks to allow `LIKE` optimisation with international character support.
- Upgraded `DBI` to 0.93.

D.5.11 Changes in release 3.21.23

- The following symbols are now reserved words: `TIME`, `DATE`, `TIMESTAMP`, `TEXT`, `BIT`, `ENUM`, `NO`, `ACTION`, `CHECK`, `YEAR`, `MONTH`, `DAY`, `HOUR`, `MINUTE`, `SECOND`, `STATUS`, `VARIABLES`.
- Setting a `TIMESTAMP` to `NULL` in `LOAD DATA INFILE ...` didn't set the current time for the `TIMESTAMP`.
- Fix `BETWEEN` to recognise binary strings. Now `BETWEEN` is case-sensitive.
- Added `--skip-thread-priority` option to `mysqld`, for systems where `mysqld`'s thread scheduling doesn't work properly (BSDI 3.1).
- Added ODBC functions `DAYNAME()` and `MONTHNAME()`.
- Added function `TIME_FORMAT()`. This works like `DATE_FORMAT()`, but takes a time string (`'HH:MM:DD'`) as argument.
- Fixed unlikely(?) key optimiser bug when using `OR`s of key parts inside `AND`s.
- Added `variables` command to `mysqladmin`.
- A lot of small changes to the binary releases.
- Fixed a bug in the new protocol from MySQL Version 3.21.20.
- Changed `ALTER TABLE` to work with Windows (Windows can't rename open files). Also fixed a couple of small bugs in the Windows version.
- All standard MySQL clients are now ported to MySQL-Windows.
- MySQL can now be started as a service on NT.

D.5.12 Changes in release 3.21.22

- Starting with this version, all MySQL distributions will be configured, compiled and tested with `crash-me` and the benchmarks on the following platforms: SunOS 5.6 sun4u, SunOS 5.5.1 sun4u, SunOS 4.14 sun4c, SunOS 5.6 i86pc, Irix 6.3 mips5k, HP-UX 10.20 hppa, AIX 4.2.1 ppc, OSF/1 V4.0 alpha, FreeBSD 2.2.2 i86pc and BSDI 3.1 i386.
- Fix `COUNT(*)` problems when the `WHERE` clause didn't match any records. (Bug from 3.21.17.)
- Removed that `NULL = NULL` is true. Now you must use `IS NULL` or `IS NOT NULL` to test whether a value is `NULL`. (This is according to ANSI SQL but may break old applications that are ported from `mSQL`.) You can get the old behaviour by compiling with `-DmSQL_COMPLIANT`.
- Fixed bug that core dumped when using many `LEFT OUTER JOIN` clauses.
- Fixed bug in `ORDER BY` on string formula with possible `NULL` values.
- Fixed problem in range optimiser when using `<=` on sub index.
- Added functions `DAYOFYEAR()`, `DAYOFMONTH()`, `MONTH()`, `YEAR()`, `WEEK()`, `QUARTER()`, `HOUR()`, `MINUTE()`, `SECOND()` and `FIND_IN_SET()`.
- Added `SHOW VARIABLES` command.
- Added support of "long constant strings" from ANSI SQL:

```
mysql> SELECT 'first ' 'second';       -> 'first second'
```

- Upgraded Msql-Mysql-modules to 1.1825.
- Upgraded `mysqlaccess` to 2.02.
- Fixed problem with Russian character set and `LIKE`.
- Ported to OpenBSD 2.1.
- New Dutch error messages.

D.5.13 Changes in release 3.21.21a

- Configure changes for some operating systems.

D.5.14 Changes in release 3.21.21

- Fixed optimiser bug when using `WHERE data_field = date_field2 AND date_field2 = constant`.
- Added `SHOW STATUS` command.
- Removed '`manual.ps`' from the source distribution to make it smaller.

D.5.15 Changes in release 3.21.20

- Changed the maximum table name and column name lengths from 32 to 64.
- Aliases can now be of "any" length.
- Fixed `mysqladmin stat` to return the right number of queries.

- Changed protocol (downward compatible) to mark if a column has the `AUTO_INCREMENT` attribute or is a `TIMESTAMP`. This is needed for the new Java driver.
- Added Hebrew sorting order by Zeev Suraski.
- Solaris 2.6: Fixed `configure` bugs and increased maximum table size from 2G to 4G.

D.5.16 Changes in release 3.21.19

- Upgraded `DBD` to 1.1823. This version implements `mysql_use_result` in `DBD-Mysql`.
- Benchmarks updated for empress (by Luuk).
- Fixed a case of slow range searching.
- Configure fixes ('`Docs`' directory).
- Added function `REVERSE()` (by Zeev Suraski).

D.5.17 Changes in release 3.21.18

- Issue error message if client C functions are called in wrong order.
- Added automatic reconnect to the '`libmysql.c`' library. If a write command fails, an automatic reconnect is done.
- Small sort sets no longer use temporary files.
- Upgraded `DBI` to 0.91.
- Fixed a couple of problems with `LEFT OUTER JOIN`.
- Added `CROSS JOIN` syntax. `CROSS` is now a reserved word.
- Recoded `yacc/bison` stack allocation to be even safer and to allow MySQL to handle even bigger expressions.
- Fixed a couple of problems with the update log.
- `ORDER BY` was slow when used with key ranges.

D.5.18 Changes in release 3.21.17

- Changed documentation string of `--with-unix-socket-path` to avoid confusion.
- Added ODBC and ANSI SQL style `LEFT OUTER JOIN`.
- The following are new reserved words: `LEFT`, `NATURAL`, `USING`.
- The client library now uses the value of the environment variable `MYSQL_HOST` as the default host if it's defined.
- `SELECT col_name, SUM(expr)` now returns `NULL` for `col_name` when there are matching rows.
- Fixed problem with comparing binary strings and `BLOB`s with ASCII characters over 127.
- Fixed lock problem: when freeing a read lock on a table with multiple read locks, a thread waiting for a write lock would have been given the lock. This shouldn't affect data integrity, but could possibly make `mysqld` restart if one thread was reading data that another thread modified.

- `LIMIT offset,count` didn't work in `INSERT ... SELECT`.
- Optimised key block caching. This will be quicker than the old algorithm when using bigger key caches.

D.5.19 Changes in release 3.21.16

- Added ODBC 2.0 & 3.0 functions `POWER()`, `SPACE()`, `COT()`, `DEGREES()`, `RADIANS()`, `ROUND(2 arg)` and `TRUNCATE()`.
- **Warning: Incompatible change!** `LOCATE()` parameters were swapped according to ODBC standard. Fixed.
- Added function `TIME_TO_SEC()`.
- In some cases, default values were not used for `NOT NULL` fields.
- Timestamp wasn't always updated properly in `UPDATE SET ...` statements.
- Allow empty strings as default values for `BLOB` and `TEXT`, to be compatible with `mysqldump`.

D.5.20 Changes in release 3.21.15

- **Warning: Incompatible change!** `mysqlperl` is now from Msql-Mysql-modules. This means that `connect()` now takes `host`, `database`, `user`, `password` arguments! The old version took `host`, `database`, `password`, `user`.
- Allow `DATE '1997-01-01'`, `TIME '12:10:10'` and `TIMESTAMP '1997-01-01 12:10:10'` formats required by ANSI SQL. **Warning: Incompatible change!** This has the unfortunate side-effect that you no longer can have columns named `DATE`, `TIME` or `TIMESTAMP`. :(Old columns can still be accessed through `tablename.columnname`!)
- Changed Makefiles to hopefully work better with BSD systems. Also, '`manual.dvi`' is now included in the distribution to avoid having stupid `make` programs trying to rebuild it.
- `readline` library upgraded to version 2.1.
- A new sortorder german-1. That is a normal ISO-Latin1 with a german sort order.
- Perl `DBI/DBD` is now included in the distribution. `DBI` is now the recommended way to connect to MySQL from Perl.
- New portable benchmark suite with `DBD`, with test results from `mSQL` 2.0.3, MySQL, PostgreSQL 6.2.1 and Solid server 2.2.
- `crash-me` is now included with the benchmarks; this is a Perl program designed to find as many limits as possible in a SQL server. Tested with `mSQL`, PostgreSQL, Solid and MySQL.
- Fixed bug in range-optimiser that crashed MySQL on some queries.
- Table and column name completion for `mysql` command-line tool, by Zeev Suraski and Andi Gutmans.
- Added new command `REPLACE` that works like `INSERT` but replaces conflicting records with the new record. `REPLACE INTO TABLE ... SELECT ...` works also.
- Added new commands `CREATE DATABASE db_name` and `DROP DATABASE db_name`.

- Added `RENAME` option to `ALTER TABLE`: `ALTER TABLE name RENAME TO new_name`.
- `make_binary_distribution` now includes '`libgcc.a`' in '`libmysqlclient.a`'. This should make linking work for people who don't have `gcc`.
- Changed `net_write()` to `my_net_write()` because of a name conflict with Sybase.
- New function `DAYOFWEEK()` compatible with ODBC.
- Stack checking and `bison` memory overrun checking to make MySQL safer with weird queries.

D.5.21 Changes in release 3.21.14b

- Fixed a couple of small `configure` problems on some platforms.

D.5.22 Changes in release 3.21.14a

- Ported to SCO Openserver 5.0.4 with FSU Pthreads.
- HP-UX 10.20 should work.
- Added new function `DATE_FORMAT()`.
- Added `NOT IN`.
- Added automatic removal of 'ODBC function conversions': `{fn now() }`
- Handle ODBC 2.50.3 option flags.
- Fixed comparison of `DATE` and `TIME` values with `NULL`.
- Changed language name from germany to german to be consistent with the other language names.
- Fixed sorting problem on functions returning a `FLOAT`. Previously, the values were converted to `INT`s before sorting.
- Fixed slow sorting when sorting on key field when using `key_column=constant`.
- Sorting on calculated `DOUBLE` values sorted on integer results instead.
- `mysql` no longer requires a database argument.
- Changed the place where `HAVING` should be. According to ANSI, it should be after `GROUP BY` but before `ORDER BY`. MySQL Version 3.20 incorrectly had it last.
- Added Sybase command `USE DATABASE` to start using another database.
- Added automatic adjusting of number of connections and table cache size if the maximum number of files that can be opened is less than needed. This should fix that `mysqld` doesn't crash even if you haven't done a `ulimit -n 256` before starting `mysqld`.
- Added lots of limit checks to make it safer when running with too little memory or when doing weird queries.

D.5.23 Changes in release 3.21.13

- Added retry of interrupted reads and clearing of `errno`. This makes Linux systems much safer!
- Fixed locking bug when using many aliases on the same table in the same `SELECT`.

- Fixed bug with `LIKE` on number key.
- New error message so you can check whether the connection was lost while the command was running or whether the connection was down from the start.
- Added `--table` option to `mysql` to print in table format. Moved time and row information after query result. Added automatic reconnect of lost connections.
- Added `!=` as a synonym for `<>`.
- Added function `VERSION()` to make easier logs.
- New multi-user test '`tests/fork_test.pl`' to put some strain on the thread library.

D.5.24 Changes in release 3.21.12

- Fixed `ftruncate()` call in MIT-pthreads. This made `isamchk` destroy the '`.ISM`' files on (Free)BSD 2.x systems.
- Fixed broken `__P_` patch in MIT-pthreads.
- Many memory overrun checks. All string functions now return `NULL` if the returned string should be longer than `max_allowed_packet` bytes.
- Changed the name of the `INTERVAL` type to `ENUM`, because `INTERVAL` is used in ANSI SQL.
- In some cases, doing a `JOIN + GROUP + INTO OUTFILE`, the result wasn't grouped.
- `LIKE` with `'_'` as last character didn't work. Fixed.
- Added extended ANSI SQL `TRIM()` function.
- Added `CURTIME()`.
- Added `ENCRYPT()` function by Zeev Suraski.
- Fixed better `FOREIGN KEY` syntax skipping. New reserved words: `MATCH`, `FULL`, `PARTIAL`.
- `mysqld` now allows IP number and hostname for the `--bind-address` option.
- Added `SET CHARACTER SET cp1251_koi8` to enable conversions of data to and from the `cp1251_koi8` character set.
- Lots of changes for Windows 95 port. In theory, this version should now be easily portable to Windows 95.
- Changed the `CREATE COLUMN` syntax of `NOT NULL` columns to be after the `DEFAULT` value, as specified in the ANSI SQL standard. This will make `mysqldump` with `NOT NULL` and default values incompatible with MySQL Version 3.20.
- Added many function name aliases so the functions can be used with ODBC or ANSI SQL92 syntax.
- Fixed syntax of `ALTER TABLE tbl_name ALTER COLUMN col_name SET DEFAULT NULL`.
- Added `CHAR` and `BIT` as synonyms for `CHAR(1)`.
- Fixed core dump when updating as a user who has only `SELECT` privilege.
- `INSERT ... SELECT ... GROUP BY` didn't work in some cases. An `Invalid use of group function` error occurred.
- When using `LIMIT`, `SELECT` now always uses keys instead of record scan. This will give better performance on `SELECT` and a `WHERE` that matches many rows.
- Added Russian error messages.

D.5.25 Changes in release 3.21.11

- Configure changes.
- MySQL now works with the new thread library on BSD/OS 3.0.
- Added new group functions `BIT_OR()` and `BIT_AND()`.
- Added compatibility functions `CHECK` and `REFERENCES`. `CHECK` is now a reserved word.
- Added `ALL` option to `GRANT` for better compatibility. (`GRANT` is still a dummy function.)
- Added partly-translated Dutch error messages.
- Fixed bug in `ORDER BY` and `GROUP BY` with `NULL` columns.
- Added function `LAST_INSERT_ID()` SQL function to retrieve last `AUTO_INCREMENT` value. This is intended for clients to ODBC that can't use the `mysql_insert_id()` API function, but can be used by any client.
- Added `--flush-logs` option to `mysqladmin`.
- Added command `STATUS` to `mysql`.
- Fixed problem with `ORDER BY/GROUP BY` because of bug in `gcc`.
- Fixed problem with `INSERT ... SELECT ... GROUP BY`.

D.5.26 Changes in release 3.21.10

- New `mysqlaccess`.
- `CREATE` now supports all ODBC types and the `mSQL TEXT` type. All ODBC 2.5 functions are also supported (added `REPEAT`). This provides better portability.
- Added text types `TINYTEXT`, `TEXT`, `MEDIUMTEXT` and `LONGTEXT`. These are actually `BLOB`types, but all searching is done in case-insensitive fashion.
- All old `BLOB` fields are now `TEXT` fields. This only changes that all searching on strings is done in case-sensitive fashion. You must do an `ALTER TABLE` and change the field type to `BLOB` if you want to have tests done in case-sensitive fashion.
- Fixed some `configure` issues.
- Made the locking code a bit safer. Fixed very unlikely deadlock situation.
- Fixed a couple of bugs in the range optimiser. Now the new range benchmark `test-select` works.

D.5.27 Changes in release 3.21.9

- Added `--enable-unix-socket=pathname` option to `configure`.
- Fixed a couple of portability problems with include files.
- Fixed bug in range calculation that could return empty set when searching on multiple key with only one entry (very rare).
- Most things ported to FSU Pthreads, which should allow MySQL to run on Caldera (SCO). Voir Section 2.6.6.9 [Caldera], page 145.

D.5.28 Changes in release 3.21.8

- Works now in Solaris 2.6.
- Added handling of calculation of `SUM()` functions. For example, you can now use `SUM(column)/COUNT(column)`.
- Added handling of trigometric functions: `PI()`, `ACOS()`, `ASIN()`, `ATAN()`, `COS()`, `SIN()` and `TAN()`.
- New languages: norwegian, norwegian-ny and portuguese.
- Fixed parameter bug in `net_print()` in '`procedure.cc`'.
- Fixed a couple of memory leaks.
- Now allow also the old `SELECT ... INTO OUTFILE` syntax.
- Fixed bug with `GROUP BY` and `SELECT` on key with many values.
- `mysql_fetch_lengths()` sometimes returned incorrect lengths when you used `mysql_use_result()`. This affected at least some cases of `mysqldump --quick`.
- Fixed bug in optimisation of `WHERE const op field`.
- Fixed problem when sorting on `NULL` fields.
- Fixed a couple of 64-bit (Alpha) problems.
- Added `--pid-file=#` option to `mysqld`.
- Added date formatting to `FROM_UNIXTIME()`, originally by Zeev Suraski.
- Fixed bug in `BETWEEN` in range optimiser (did only test = of the first argument).
- Added machine-dependent files for MIT-pthreads i386-SCO. There is probably more to do to get this to work on SCO 3.5.

D.5.29 Changes in release 3.21.7

- Changed '`Makefile.am`' to take advantage of Automake 1.2.
- Added the beginnings of a benchmark suite.
- Added more secure password handling.
- Added new client function `mysql_errno()`, to get the error number of the error message. This makes error checking in the client much easier. This makes the new server incompatible with the 3.20.x server when running without `--old-protocol`. The client code is backward-compatible. More information can be found in the '`README`' file!
- Fixed some problems when using very long, illegal names.

D.5.30 Changes in release 3.21.6

- Fixed more portability issues (incorrect `sigwait` and `sigset` defines).
- `configure` should now be able to detect the last argument to `accept()`.

D.5.31 Changes in release 3.21.5

- Should now work with FreeBSD 3.0 if used with '`FreeBSD-3.0-libc_r-1.0.diff`', which can be found at `http://www.mysql.com/Downloads/Patches/`.

- Added new `-O tmp_table_size=#` option to `mysqld`.
- New function `FROM_UNIXTIME(timestamp)` which returns a date string in `'YYYY-MM-DD HH:MM:DD'` format.
- New function `SEC_TO_TIME(seconds)` which returns a string in `'HH:MM:SS'` format.
- New function `SUBSTRING_INDEX()`, originally by Zeev Suraski.

D.5.32 Changes in release 3.21.4

- Should now configure and compile on OSF/1 4.0 with the DEC compiler.
- Configuration and compilation on BSD/OS 3.0 works, but due to some bugs in BSD/OS 3.0, `mysqld` doesn't work on it yet.
- Configuration and compilation on FreeBSD 3.0 works, but I couldn't get `pthread_create` to work.

D.5.33 Changes in release 3.21.3

- Added reverse check lookup of hostnames to get better security.
- Fixed some possible buffer overflows if filenames that are too long are used.
- `mysqld` doesn't accept hostnames that start with digits followed by a `'.'`, because the hostname may look like an IP number.
- Added `--skip-networking` option to `mysqld`, to allow only socket connections. (This will not work with MIT-pthreads!)
- Added check of too long table names for alias.
- Added check if database name is okay.
- Added check if too long table names.
- Removed incorrect `free()` that killed the server on `CREATE DATABASE` or `DROP DATABASE`.
- Changed some `mysqld -O` options to better names.
- Added `-O join_cache_size=#` option to `mysqld`.
- Added `-O max_join_size=#` option to `mysqld`, to be able to set a limit how big queries (in this case big = slow) one should be able to handle without specifying `SET SQL_BIG_SELECTS=1`. A # = is about 10 examined records. The default is "unlimited".
- When comparing a `TIME`, `DATE`, `DATETIME` or `TIMESTAMP` column to a constant, the constant is converted to a time value before performing the comparison. This will make it easier to get ODBC (particularly Access97) to work with the above types. It should also make dates easier to use and the comparisons should be quicker than before.
- Applied patch from Jochen Wiedmann that allows `query()` in `mysqlperl` to take a query with `\0` in it.
- Storing a timestamp with a 2-digit year (`YYMMDD`) didn't work.
- Fix that timestamp wasn't automatically updated if set in an `UPDATE` clause.
- Now the automatic timestamp field is the FIRST timestamp field.
- `SELECT * INTO OUTFILE`, which didn't correctly if the outfile already existed.

- `mysql` now shows the thread ID when starting or doing a reconnect.
- Changed the default sort buffer size from 2M to 1M.

D.5.34 Changes in release 3.21.2

- The range optimiser is coded, but only 85% tested. It can be enabled with `--new`, but it crashes core a lot yet...
- More portable. Should compile on AIX and alpha-digital. At least the `isam` library should be relatively 64-bit clean.
- New `isamchk` which can detect and fix more problems.
- New options for `isamlog`.
- Using new version of Automake.
- Many small portability changes (from the AIX and alpha-digital port) Better checking of pthread(s) library.
- czech error messages by `snajdr@pvt.net`.
- Decreased size of some buffers to get fewer problems on systems with little memory. Also added more checks to handle "out of memory" problems.
- `mysqladmin`: you can now do `mysqladmin kill 5,6,7,8` to kill multiple threads.
- When the maximum connection limit is reached, one extra connection by a user with the **process_acl** privilege is granted.
- Added `-O backlog=#` option to `mysqld`.
- Increased maximum packet size from 512K to 1024K for client.
- Almost all of the function code is now tested in the internal test suite.
- `ALTER TABLE` now returns warnings from field conversions.
- Port changed to 3306 (got it reserved from ISI).
- Added a fix for Visual FoxBase so that any schema name from a table specification is automatically removed.
- New function `ASCII()`.
- Removed function `BETWEEN(a,b,c)`. Use the standard ANSI syntax instead: `expr BETWEEN expr AND expr`.
- MySQL no longer has to use an extra temporary table when sorting on functions or `SUM()` functions.
- Fixed bug that you couldn't use `tbl_name.field_name` in `UPDATE`.
- Fixed `SELECT DISTINCT` when using 'hidden group'. For example:

```
mysql> SELECT DISTINCT MOD(some_field,10) FROM test
    ->        GROUP BY some_field;
```

  Note: `some_field` is normally in the `SELECT` part. ANSI SQL should require it.

D.5.35 Changes in release 3.21.0

- New reserved words used: `INTERVAL`, `EXPLAIN`, `READ`, `WRITE`, `BINARY`.
- Added ODBC function `CHAR(num,...)`.

- New operator `IN`. This uses a binary search to find a match.
- New command `LOCK TABLES tbl_name [AS alias] {READ|WRITE} ...`
- Added `--log-update` option to `mysqld`, to get a log suitable for incremental updates.
- New command `EXPLAIN SELECT ...` to get information about how the optimiser will do the join.
- For easier client code, the client should no longer use `FIELD_TYPE_TINY_BLOB`, `FIELD_TYPE_MEDIUM_BLOB`, `FIELD_TYPE_LONG_BLOB` or `FIELD_TYPE_VAR_STRING` (as previously returned by `mysql_list_fields`). You should instead only use `FIELD_TYPE_BLOB` or `FIELD_TYPE_STRING`. If you want exact types, you should use the command `SHOW FIELDS`.
- Added varbinary syntax: `0x######` which can be used as a string (default) or a number.
- `FIELD_TYPE_CHAR` is renamed to `FIELD_TYPE_TINY`.
- Changed all fields to C++ classes.
- Removed FORM struct.
- Fields with `DEFAULT` values no longer need to be `NOT NULL`.
- New field types:

  `ENUM` A string which can take only a couple of defined values. The value is stored as a 1-3 byte number that is mapped automatically to a string. This is sorted according to string positions!

  `SET` A string which may have one or many string values separated with ','. The string is stored as a 1-, 2-, 3-, 4- or 8-byte number where each bit stands for a specific set member. This is sorted according to the unsigned value of the stored packed number.
- Now all function calculation is done with `double` or `long long`. This will provide the full 64-bit range with bit functions and fix some conversions that previously could result in precision losses. One should avoid using `unsigned long long` columns with full 64-bit range (numbers bigger than 9223372036854775807) because calculations are done with `signed long long`.
- `ORDER BY` will now put `NULL` field values first. `GROUP BY` will also work with `NULL` values.
- Full `WHERE` with expressions.
- New range optimiser that can resolve ranges when some keypart prefix is constant. Example:

```
mysql> SELECT * FROM tbl_name
    ->          WHERE key_part_1="customer"
    ->          AND key_part_2>=10 AND key_part_2<=10;
```

D.6 Changes in release 3.20.x

Version 3.20 is quite old now, and should be avoided if possible. This information is kept here for historical purposes only.

Changes from 3.20.18 to 3.20.32b are not documented here because the 3.21 release branched here. And the relevant changes are also documented as changes to the 3.21 version.

D.6.1 Changes in release 3.20.18

- Added `-p#` (remove `#` directories from path) to `isamlog`. All files are written with a relative path from the database directory Now `mysqld` shouldn't crash on shutdown when using the `--log-isam` option.
- New `mysqlperl` version. It is now compatible with `msqlperl-0.63`.
- New `DBD` module available at `http://www.mysql.com/Downloads/Contrib/` site.
- Added group function `STD()` (standard deviation).
- The `mysqld` server is now compiled by default without debugging information. This will make the daemon smaller and faster.
- Now one usually only has to specify the `--basedir` option to `mysqld`. All other paths are relative in a normal installation.
- `BLOB` columns sometimes contained garbage when used with a `SELECT` on more than one table and `ORDER BY`.
- Fixed that calculations that are not in `GROUP BY` work as expected (ANSI SQL extension). Example:

```
mysql> SELECT id,id+1 FROM table GROUP BY id;
```

- The test of using `MYSQL_PWD` was reversed. Now `MYSQL_PWD` is enabled as default in the default release.
- Fixed conversion bug which caused `mysqld` to core dump with Arithmetic error on SPARC-386.
- Added `--unbuffered` option to `mysql`, for new `mysqlaccess`.
- When using overlapping (unnecessary) keys and join over many tables, the optimiser could get confused and return 0 records.

D.6.2 Changes in release 3.20.17

- You can now use `BLOB` columns and the functions `IS NULL` and `IS NOT NULL` in the `WHERE` clause.
- All communication packets and row buffers are now allocated dynamically on demand. The default value of `max_allowed_packet` is now 64K for the server and 512K for the client. This is mainly used to catch incorrect packets that could trash all memory. The server limit may be changed when it is started.
- Changed stack usage to use less memory.
- Changed `safe_mysqld` to check for running daemon.
- The `ELT()` function is renamed to `FIELD()`. The new `ELT()` function returns a value based on an index: `FIELD()` is the inverse of `ELT()` Example: `ELT(2,"A","B","C")` returns `"B"`. `FIELD("B","A","B","C")` returns 2.
- `COUNT(field)`, where `field` could have a `NULL` value, now works.
- A couple of bugs fixed in `SELECT ... GROUP BY`.
- Fixed memory overrun bug in `WHERE` with many unoptimisable brace levels.
- Fixed some small bugs in the grant code.

- If hostname isn't found by `get_hostname`, only the IP is checked. Previously, you got `Access denied`.
- Inserts of timestamps with values didn't always work.
- `INSERT INTO ... SELECT ... WHERE` could give the error `Duplicated field`.
- Added some tests to `safe_mysqld` to make it "safer".
- `LIKE` was case-sensitive in some places and case-insensitive in others. Now `LIKE` is always case-insensitive.
- '`mysql.cc`': Allow `'#'` anywhere on the line.
- New command `SET SQL_SELECT_LIMIT=#`. See the FAQ for more details.
- New version of the `mysqlaccess` script.
- Change `FROM_DAYS()` and `WEEKDAY()` to also take a full `TIMESTAMP` or `DATETIME` as argument. Before they only took a number of type `YYYYMMDD` or `YYMMDD`.
- Added new function `UNIX_TIMESTAMP(timestamp_column)`.

D.6.3 Changes in release 3.20.16

- More changes in MIT-pthreads to get them safer. Fixed also some link bugs at least in SunOS.
- Changed `mysqld` to work around a bug in MIT-pthreads. This makes multiple small `SELECT` operations 20 times faster. Now `lock_test.pl` should work.
- Added `mysql_FetchHash(handle)` to `mysqlperl`.
- The `mysqlbug` script is now distributed built to allow for reporting bugs that appear during the build with it.
- Changed '`libmysql.c`' to prefer `getpwuid()` instead of `cuserid()`.
- Fixed bug in `SELECT` optimiser when using many tables with the same column used as key to different tables.
- Added new `latin2` and Russian `KOI8` character tables.
- Added support for a dummy `GRANT` command to satisfy Powerbuilder.

D.6.4 Changes in release 3.20.15

- Fixed fatal bug `packets out of order` when using MIT-pthreads.
- Removed possible loop when a thread waits for command from client and `fcntl()` fails. Thanks to Mike Bretz for finding this bug.
- Changed alarm loop in '`mysqld.cc`' because shutdown didn't always succeed in Linux.
- Removed use of `termbits` from '`mysql.cc`'. This conflicted with `glibc` 2.0.
- Fixed some syntax errors for at least BSD and Linux.
- Fixed bug when doing a `SELECT` as superuser without a database.
- Fixed bug when doing `SELECT` with group calculation to outfile.

D.6.5 Changes in release 3.20.14

- If one gives `-p` or `--password` option to `mysql` without an argument, the user is solicited for the password from the tty.
- Added default password from `MYSQL_PWD` (by Elmar Haneke).
- Added command `kill` to `mysqladmin` to kill a specific MySQL thread.
- Sometimes when doing a reconnect on a down connection this succeeded first on second try.
- Fixed adding an `AUTO_INCREMENT` key with `ALTER_TABLE`.
- `AVG()` gave too small value on some `SELECT`s with `GROUP BY` and `ORDER BY`.
- Added new `DATETIME` type (by Giovanni Maruzzelli `maruzz@matrice.it`).
- Fixed that defining `DONT_USE_DEFAULT_FIELDS` works.
- Changed to use a thread to handle alarms instead of signals on Solaris to avoid race conditions.
- Fixed default length of signed numbers. (George Harvey `georgeh@pinacl.co.uk`.)
- Allow anything for `CREATE INDEX`.
- Add prezeros when packing numbers to `DATE`, `TIME` and `TIMESTAMP`.
- Fixed a bug in `OR` of multiple tables (gave empty set).
- Added many patches to MIT-pthreads. This fixes at least one lookup bug.

D.6.6 Changes in release 3.20.13

- Added ANSI SQL94 `DATE` and `TIME` types.
- Fixed bug in `SELECT` with `AND-OR` levels.
- Added support for Slovenian characters. The '`Contrib`' directory contains source and instructions for adding other character sets.
- Fixed bug with `LIMIT` and `ORDER BY`.
- Allow `ORDER BY` and `GROUP BY` on items that aren't in the `SELECT` list. (Thanks to Wim Bonis `bonis@kiss.de`, for pointing this out.)
- Allow setting of timestamp values in `INSERT`.
- Fixed bug with `SELECT ... WHERE ... = NULL`.
- Added changes for `glibc` 2.0. To get `glibc` to work, you should add the '`gibc-2.0-sigwait-patch`' before compiling `glibc`.
- Fixed bug in `ALTER TABLE` when changing a `NOT NULL` field to allow `NULL` values.
- Added some ANSI92 synonyms as field types to `CREATE TABLE`. `CREATE TABLE` now allows `FLOAT(4)` and `FLOAT(8)` to mean `FLOAT` and `DOUBLE`.
- New utility program `mysqlaccess` by `Yves.Carlier@rug.ac.be`. This program shows the access rights for a specific user and the grant rows that determine this grant.
- Added `WHERE const op field` (by `bonis@kiss.de`).

D.6.7 Changes in release 3.20.11

- When using `SELECT ... INTO OUTFILE`, all temporary tables are ISAM instead of HEAP to allow big dumps.
- Changed date functions to be string functions. This fixed some "funny" side effects when sorting on dates.
- Extended `ALTER TABLE` according to SQL92.
- Some minor compatibility changes.
- Added `--port` and `--socket` options to all utility programs and `mysqld`.
- Fixed MIT-pthreads `readdir_r()`. Now `mysqladmin create database` and `mysqladmin drop database` should work.
- Changed MIT-pthreads to use our `tempnam()`. This should fix the "sort aborted" bug.
- Added sync of records count in `sql_update`. This fixed slow updates on first connection. (Thanks to Vaclav Bittner for the test.)

D.6.8 Changes in release 3.20.10

- New insert type: `INSERT INTO ... SELECT ...`
- `MEDIUMBLOB` fixed.
- Fixed bug in `ALTER TABLE` and `BLOB`s.
- `SELECT ... INTO OUTFILE` now creates the file in the current database directory.
- `DROP TABLE` now can take a list of tables.
- Oracle synonym `DESCRIBE` (`DESC`).
- Changes to `make_binary_distribution`.
- Added some comments to installation instructions about `configure`'s C++ link test.
- Added `--without-perl` option to `configure`.
- Lots of small portability changes.

D.6.9 Changes in release 3.20.9

- `ALTER TABLE` didn't copy null bit. As a result, fields that were allowed to have `NULL` values were always `NULL`.
- `CREATE` didn't take numbers as `DEFAULT`.
- Some compatibility changes for SunOS.
- Removed '`config.cache`' from old distribution.

D.6.10 Changes in release 3.20.8

- Fixed bug with `ALTER TABLE` and multi-part keys.

D.6.11 Changes in release 3.20.7

- New commands: `ALTER TABLE`, `SELECT ... INTO OUTFILE` and `LOAD DATA INFILE`.
- New function: `NOW()`.
- Added new field `File_priv` to `mysql/user` table.
- New script `add_file_priv` which adds the new field `File_priv` to the `user` table. This script must be executed if you want to use the new `SELECT ... INTO` and `LOAD DATA INFILE ...` commands with a version of MySQL earlier than 3.20.7.
- Fixed bug in locking code, which made `lock_test.pl` test fail.
- New files '`NEW`' and '`BUGS`'.
- Changed '`select_test.c`' and '`insert_test.c`' to include '`config.h`'.
- Added `status` command to `mysqladmin` for short logging.
- Increased maximum number of keys to 16 and maximum number of key parts to 15.
- Use of sub keys. A key may now be a prefix of a string field.
- Added `-k` option to `mysqlshow`, to get key information for a table.
- Added long options to `mysqldump`.

D.6.12 Changes in release 3.20.6

- Portable to more systems because of MIT-pthreads, which will be used automatically if `configure` cannot find a `-lpthreads` library.
- Added GNU-style long options to almost all programs. Test with *`program`* `--help`.
- Some shared library support for Linux.
- The FAQ is now in '`.texi`' format and is available in '`.html`', '`.txt`' and '`.ps`' formats.
- Added new SQL function `RAND([init])`.
- Changed `sql_lex` to handle `\0` unquoted, but the client can't send the query through the C API, because it takes a str pointer. You must use `mysql_real_query()` to send the query.
- Added API function `mysql_get_client_info()`.
- `mysqld` now uses the `N_MAX_KEY_LENGTH` from '`nisam.h`' as the maximum allowable key length.
- The following now works:

  ```
  mysql> SELECT filter_nr,filter_nr FROM filter ORDER BY filter_nr;
  ```

  Previously, this resulted in the error: `Column: 'filter_nr' in order clause is ambiguous`.
- `mysql` now outputs `'\0'`, `'\t'`, `'\n'` and `'\\'` when encountering ASCII 0, tab, newline or `'\'` while writing tab-separated output. This is to allow printing of binary data in a portable format. To get the old behaviour, use `-r` (or `--raw`).
- Added german error messages (60 of 80 error messages translated).
- Added new API function `mysql_fetch_lengths(MYSQL_RES *)`, which returns an array of column lengths (of type `uint`).

- Fixed bug with `IS NULL` in `WHERE` clause.
- Changed the optimiser a little to get better results when searching on a key part.
- Added `SELECT` option `STRAIGHT_JOIN` to tell the optimiser that it should join tables in the given order.
- Added support for comments starting with `'--'` in '`mysql.cc`' (Postgres syntax).
- You can have `SELECT` expressions and table columns in a `SELECT` which are not used in the group part. This makes it efficient to implement lookups. The column that is used should be a constant for each group because the value is calculated only once for the first row that is found for a group.

```
mysql> SELECT id,lookup.text,SUM(*) FROM test,lookup
    ->        WHERE test.id=lookup.id GROUP BY id;
```

- Fixed bug in `SUM(function)` (could cause a core dump).
- Changed `AUTO_INCREMENT` placement in the SQL query:

```
INSERT INTO table (auto_field) VALUES (0);
```

  inserted 0, but it should insert an `AUTO_INCREMENT` value.
- '`mysqlshow.c`': Added number of records in table. Had to change the client code a little to fix this.
- `mysql` now allows doubled `''` or `""` within strings for embedded `'` or `"`.
- New math functions: `EXP()`, `LOG()`, `SQRT()`, `ROUND()`, `CEILING()`.

D.6.13 Changes in release 3.20.3

- The `configure` source now compiles a thread-free client library `-lmysqlclient`. This is the only library that needs to be linked with client applications. When using the binary releases, you must link with `-lmysql -lmysys -ldbug -lmystrings` as before.
- New `readline` library from `bash-2.0`.
- LOTS of small changes to `configure` and makefiles (and related source).
- It should now be possible to compile in another directory using `VPATH`. Tested with GNU Make 3.75.
- `safe_mysqld` and `mysql.server` changed to be more compatible between the source and the binary releases.
- `LIMIT` now takes one or two numeric arguments. If one argument is given, it indicates the maximum number of rows in a result. If two arguments are given, the first argument indicates the offset of the first row to return, the second is the maximum number of rows. With this it's easy to do a poor man's next page/previous page WWW application.
- Changed name of SQL function `FIELDS()` to `ELT()`. Changed SQL function `INTERVALL()` to `INTERVAL()`.
- Made `SHOW COLUMNS` a synonym for `SHOW FIELDS`. Added compatibility syntax `FRIEND KEY` to `CREATE TABLE`. In MySQL, this creates a non-unique key on the given columns.
- Added `CREATE INDEX` and `DROP INDEX` as compatibility functions. In MySQL, `CREATE INDEX` only checks if the index exists and issues an error if it doesn't exist. `DROP INDEX` always succeeds.

- '`mysqladmin.c`': added client version to version information.
- Fixed core dump bug in `sql_acl` (core on new connection).
- Removed `host`, `user` and `db` tables from database `test` in the distribution.
- `FIELD_TYPE_CHAR` can now be signed (-128 to 127) or unsigned (0 to 255) Previously, it was always unsigned.
- Bug fixes in `CONCAT()` and `WEEKDAY()`.
- Changed a lot of source to get `mysqld` to be compiled with SunPro compiler.
- SQL functions must now have a `'('` immediately after the function name (no intervening space). For example, `'USER('` is regarded as beginning a function call, and `'USER ('` is regarded as an identifier `USER` followed by a `'('`, not as a function call.

D.6.14 Changes in release 3.20.0

- The source distribution is done with `configure` and Automake. It will make porting much easier. The `readline` library is included in the distribution.
- Separate client compilation: the client code should be very easy to compile on systems which don't have threads.
- The old Perl interface code is automatically compiled and installed. Automatic compiling of `DBD` will follow when the new `DBD` code is ported.
- Dynamic language support: `mysqld` can now be started with Swedish or English (default) error messages.
- New functions: `INSERT()`, `RTRIM()`, `LTRIM()` and `FORMAT()`.
- `mysqldump` now works correctly for all field types (even `AUTO_INCREMENT`). The format for `SHOW FIELDS FROM tbl_name` is changed so the `Type` column contains information suitable for `CREATE TABLE`. In previous releases, some `CREATE TABLE` information had to be patched when re-creating tables.
- Some parser bugs from 3.19.5 (`BLOB` and `TIMESTAMP`) are corrected. `TIMESTAMP` now returns different date information depending on its create length.
- Changed parser to allow a database, table or field name to start with a number or `'_'`.
- All old C code from Unireg changed to C++ and cleaned up. This makes the daemon a little smaller and easier to understand.
- A lot of small bug fixes done.
- New '`INSTALL`' files (not final version) and some information regarding porting.

D.7 Changes in release 3.19.x

Version 3.19 is quite old now, and should be avoided if possible. This information is kept here for historical purposes only.

D.7.1 Changes in release 3.19.5

- Some new functions, some more optimisation on joins.
- Should now compile clean on Linux (2.0.x).

- Added functions `DATABASE()`, `USER()`, `POW()`, `LOG10()` (needed for ODBC).
- In a `WHERE` with an `ORDER BY` on fields from only one table, the table is now preferred as first table in a multi-join.
- `HAVING` and `IS NULL` or `IS NOT NULL` now works.
- A group on one column and a sort on a group function (`SUM()`, `AVG()`...) didn't work together. Fixed.
- `mysqldump`: Didn't send password to server.

D.7.2 Changes in release 3.19.4

- Fixed horrible locking bug when inserting in one thread and reading in another thread.
- Fixed one-off decimal bug. 1.00 was output as 1.0.
- Added attribute `'Locked'` to process list as info if a query is locked by another query.
- Fixed full magic timestamp. Timestamp length may now be 14, 12, 10, 8, 6, 4 or 2 bytes.
- Sort on some numeric functions could sort incorrectly on last number.
- `IF(arg,syntax_error,syntax_error)` crashed.
- Added functions `CEILING()`, `ROUND()`, `EXP()`, `LOG()` and `SQRT()`.
- Enhanced `BETWEEN` to handle strings.

D.7.3 Changes in release 3.19.3

- Fixed `SELECT` with grouping on `BLOB` columns not to return incorrect `BLOB` info. Grouping, sorting and distinct on `BLOB` columns will not yet work as expected (probably it will group/sort by the first 7 characters in the `BLOB`). Grouping on formulas with a fixed string size (use `MID()` on a `BLOB`) should work.
- When doing a full join (no direct keys) on multiple tables with `BLOB` fields, the `BLOB` was garbage on output.
- Fixed `DISTINCT` with calculated columns.

Annexe E Port vers d'autres systémes

Cet appendice vous aidera á porter MySQL vers un autre systéme d'exploitation. Vèrifiez d'abord la liste des systémes supportès avant toute chose. Voir Section 2.2.3 [Which OS], page 74. Si vous avez crèe un nouveau port de MySQL, merci de nous en avertir pour que nous puissions le lister ici et sur notre site web (`http://www.mysql.com/`), pour le recommander aux autres utilisateurs.

Note : Si vous crèez un nouveau port de MySQL, vous êtes libre de le copier et le distribuer sous la licence GPL, mais cela ne signifie pas que vous êtes dètenteur de droits sur MySQL.

Une librairie de threads Posix qui fonctionne est requise pour le serveur. Pour Solaris 2.5 nous utilisons Sun PThreads (le support natif des threads de la version 2.4 et plus ancienne n'est pas assez bonne) et sur Linux nous utilisons LinuxThreads de Xavier Leroy, `Xavier.Leroy@inria.fr`.

La partie la plus difficile du port vers une nouvelle variante Unix ne bènèficiant pas d'un bon support natif des threads est probablement le port de MIT-pthreads. Voyez '`mit-pthreads/README`' et Programming POSIX Threads (`http://www.humanfactor.com/pthreads/`).

La distribution MySQL inclut une version patchèe des Pthreads de Provenzano de MIT (voyez la page web des Pthreads MIT `http://www.mit.edu:8001/people/proven/pthreads.html`). Cela peut être utilisè pour certains systémes d'exploitation qui n'ont pas les threads POSIX.

Il est aussi possible d'utiliser un autre package de threads au niveau utilisateur nommè FSU Pthreads (Voir la page de FSU Pthreads (`http://www.informatik.hu-berlin.de/~mueller/pthreads.ht`
Cette implèmentation est utilisèe pour le port vers SCO.

Consultez les programmes '`thr_lock.c`' et '`thr_alarm.c`' dans le dossier '`mysys`' pour quelques tests/exemples de ces problémes.

Le serveur et le client on besoin d'un compilateur C++ fonctionnel (nous utilisons `gcc` et avons essayè SPARCworks). Un autre compilateur connu maintenant pour fonctionner est Irix `cc`.

Pour ne compiler que le client, utilisez `./configure --without-server`.

Il n'y a actuellement aucun support pour ne compiler que le serveur, et il n'est pas prèvu d'en ajouter un á moins que quelqu'un n'ait une bonne raison de le faire.

Si vous voulez ou avez besoin de changer un fichier '`Makefile`' ou le script de configuration vous aurez besoin d'avoir Automake et Autoconf. Nous avons utilisè les distributions `automake-1.2` et `autoconf-2.12`.

Toutes les ètapes dont vous avez besoin pour reconstruire le tout á partir des fichiers de base.

```
/bin/rm */.deps/*.P
/bin/rm -f config.cache
aclocal
autoheader
aclocal
automake
autoconf
```

```
./configure --with-debug=full --prefix='votre dossier installation'

# les fichiers make gènèrès plus haut ont besoin de GNU make 3.75 ou plus rècent.
# (appelè gmake ci-dessous)
gmake clean all install init-db
```

Si vous rencontrez des problémes avec un nouveau port, vois devrez faire du dèbogage de MySQL ! Voir Section E.1 [Debugging server], page 815.

Note : avant de commencer á dèboguer `mysqld`, faites d'abord fonctionner les programmes de tests `mysys/thr_alarm` et `mysys/thr_lock`. Cela assurera que votre installation des threads a une chance de fonctionner !

E.1 Dèboguer un serveur MySQL

Si vous utilisez des fonctionnalitès qui ont ètè ajoutèes il y'a peu de temps á MySQL, vous pouvez essayer de dèmarrer `mysqld` avec `--skip-new` (qui dèsactivera toutes les fonctionnalitès nouvelles, qui sont potentiellement non-stables) ou avec `--safe-mode` qui dèsactive un tas d'optimisations qui pourraient poser problémes. Voir Section A.4.1 [Crashing], page 697.

Si `mysqld` ne veut pas dèmarrer, vous devez vèrifier que vous n'avez pas de fichiers '`my.cnf`' qui interférent avec votre configuration ! Vous pouvez vèrifier les arguments de votre '`my.cnf`' avec `mysqld --print-defaults` et èviter de les utiliser en dèmarrant avec `mysqld --no-defaults ...`.

Si `mysqld` se met á trop consommer de mèmoire ou de CPU ou si il se bloque, vous pouvez utiliser `mysqladmin processlist status` pour trouver si quelqu'un utilise une requête qui prend trop de temps á s'exècuter. C'est un bonne idèe d'exècuter `mysqladmin -i10 processlist status` dans un terminal si vous avez des problémes de performances ou des problémes á la connexion de nouveaux clients.

La commande `mysqladmin debug` ècrira des informations á propos des verrous en cours d'utilisation, de la mèmoire utilisèe et des requêtes dans le fichier de log de MySQL. Cela peut vous aider á rèsoudre certains problémes. Cette commande fournit aussi des informations utiles même si vous n'avez pas compilè MySQL pour le dèbogage !

Si le probléme vient du fait que certaines tables sont de plus en plus lentes vous devez essayer de les optimiser en utilisant `OPTIMIZE TABLE` ou `myisamchk`. Voir Chapitre 4 [MySQL Database Administration], page 192. Vous devez aussi vèrifier les requêtes qui prennent trop de temps avec la commande `EXPLAIN`.

Vous devriez aussi consulter les sections spècifiques aux systémes d'exploitations dans ce manuel pour les problémes pouvant être uniques á votre environnement. Voir Section 2.6 [Operating System Specific Notes], page 113.

E.1.1 Compiler MYSQL pour le dèbogage

Si vous avez un probléme spècifique, vous pouvez toujours essayer de dèboguer MySQL. Pour ce faire, vous devez configurer MySQL avec l'option `--with-debug` ou `--with-debug=full`. Vous pouvez vèrifier si MySQL est dèjá compilè avec le support du dèbogage en faisant ceci :

`mysqld --help`. Si l'attribut `--debug` est listè avec les options, cela veut dire que le support du dèbogage est activè. Dans ce cas, `mysqladmin ver` liste aussi la version de `mysqld` en tant que `mysql ... --debug`.

Si vous utilisez gcc ou egcs, la ligne de configuration recommandèe est :

```
CC=gcc CFLAGS="-O2" CXX=gcc CXXFLAGS="-O2 -felide-constructors \
   -fno-exceptions -fno-rtti" ./configure --prefix=/usr/local/mysql \
   --with-debug --with-extra-charsets=complex
```

Cela èvitera les problémes avec la librairie `libstdc++` et les exceptions C++ (plusieurs compilateurs ont des problémes avec les exceptions C++ dans le code threadè) et compilera une version MySQL avec le support de tous les jeux de caractéres.

Si vous suspectez un dèpassement de la mèmoire, vous pouvez configurer MySQL avec `--with-debug=full`, qui installera un vèrificateur d'allocation mèmoire (`SAFEMALLOC`). Fonctionner avec `SAFEMALLOC` est cependant un peu ralentissant, et donc, si vous rencontrez des problémes de performances, vous devez dèmarrer `mysqld` avec l'option `--skip-safemalloc`. Cela dèsactivera les vèrifications de dèpassements de mèmoire pour chaque appel á `malloc` ou `free`.

Si `mysqld` ne crashe plus lorsque vous le compilez avec `--with-debug`, vous avez probablement trouvè un bogue du compilateur ou un bogue de temporisation dans MySQL. Dans ce cas, vous pouvez essayer d'ajouter `-g` aux variables `CFLAGS` et `CXXFLAGS` vues plus haut et ne pas utiliser `--with-debug`. Si `mysqld` crashe maintenant, vous pouvez vous y attacher avec `gdb` ou utiliser `gdb` sur le fichier noyau pour trouver ce qui est arrivè.

Lorsque vous configurez MySQL pour le support du dèbogage, vous activez automatiquement un tas de fonctions de tests supplèmentaires qui se chargent de surveiller le bon fonctionnement de `mysqld`. Si elles trouvent quelque chose d'inattendu ("unexpected"), une entrèe sera ècrite dans `stderr`, que `safe_mysqld` redirige vers le log d'erreurs ! Cela signifie aussi que si vous avez quelques problémes inattendus avec MySQL et que vous utilisez une distribution des sources, la premiére chose á faire est de configurer MySQL avec le support du dèbogage ! (la seconde, bien sûr, ètant d'envoyer un mail á `mysql@lists.mysql.com` et demander de l'aide. Merci d'utiliser le script `mysqlbug` pour tous les rapport de bogues ou questions concernant la version de MySQL que vous utilisez !

Dans la distribution Windows de MySQL, `mysqld.exe` est par dèfaut compilè avec le support des fichiers de traçage.

E.1.2 Crèer un fichier de traçage

Si le serveur `mysqld` ne dèmarre pas ou que vous pouvez le crasher facilement, vous pouvez essayer de crèer un fichier de traçage pour trouver le probléme.

Pour ce faire, vous devez avoir un `mysqld` qui est compilè pour le dèbogage. Vous pouvez le vèrifier en exècutant `mysqld -V`. Si le numèro de version se termine par `-debug`, il est compilè avec le support des fichiers de traçage.

Dèmarrez le serveur `mysqld` avec un journal de suivi dans '`/tmp/mysqld.trace`' (ou '`C:\mysqld.trace`' sous Windows) :

```
mysqld --debug
```

Sous Windows vous devez aussi utiliser l'option `--standalone` pour ne pas dèmarrer `mysqld` en tant que service :

Dans une console DOS entrez :

```
mysqld --debug --standalone
```

Aprés cela, vous pouvez utiliser l'outil en ligne de commande `mysql.exe` dans une seconde fenêtre pour reproduire le probléme. Vous pouvez couper le serveur avec la commande `mysqladmin shutdown`.

Notez que le fichier de traçage deviendra **trés gros** ! si vous voulez obtenir un fichier plus petit, utilisez ce qui suit par exemple :

```
mysqld --debug=d,info,error,query,general,where:0,/tmp/mysqld.trace
```

qui n'ècrit que les informations les plus intèressantes dans '`/tmp/mysqld.trace`'.

Si vous crèez un rapport de bogue, merci de n'envoyer que les lignes du fichier de traçage oú le probléme se concrètise á la liste de diffusion appropriée ! Si vous n'arrivez pas á trouver le bon endroit dans le fichier, vous pouvez envoyer la totalitè du fichier ainsi que le rapport de bogue via FTP á `ftp://support.mysql.com/pub/mysql/secret/` pour qu'un dèveloppeur MySQL y jette un coup d'oeil.

Le fichier de traçage est gènèrè avec le package **DBUG** de Fred Fish. Voir Section E.3 [The DBUG package], page 821.

E.1.3 Dèboguer mysqld sous gdb

Sur la plupart des systémes, vous pouvez dèmarrer `mysqld` á partir de `gdb` pour obtenir plus d'informations si `mysqld` crashe.

Avec quelques anciennes versions de `gdb` sous Linux vous devez exècuter `run --one-thread` si vous voulez être capables de dèboguer les threads de `mysqld` threads. Dans ce cas, vous ne pouvez n'avoir qu'un thread actif á la fois. Nous vous recommandons de mettre á jour gdb á la version 5.1 dés que possible vu que le dèbogage des threads fonctionne mieux avec cette version !

Lors de l'utilisation de `mysqld` sous gdb, vous devez dèsactiver le traçage de la pile avec `--skip-stack-trace` pour pouvoir trouver les erreurs de segmentations avec gdb.

Il est trés difficile de dèboguer MySQL sous `gdb` si vous effectuez plusieurs nouvelles connexions tout le temps vu que `gdb` ne libére pas la mèmoire occupèe par les anciens threads. Vous pouvez contourner ce probléme en dèmarrant `mysqld` avec `-O thread_cache_size='max_connections +1'`. Dans la plupart des cas, le simple fait d'utiliser `-O thread_cache_size=5'` vous aidera beaucoup !

Si vous voulez obtenir un core dump sur Linux si `mysqld` se termine avec un signal SIGSEGV, vous pouvez dèmarrer `mysqld` avec l'option `--core-file`. Ce fichier noyau peut être utilisè pour effectuer des traçages qui peuvent vous aider á trouver pourquoi `mysqld` s'est terminèe :

```
shell> gdb mysqld core
gdb>   backtrace full
gdb>   exit
```

Voir Section A.4.1 [Crashing], page 697.

Si vous utilisez gdb 4.17.x ou plus rècent sous Linux, vous devez installer un fichier '`.gdb`', avec les informations suivantes, dans votre rèpertoire courant :

```
set print sevenbit off
handle SIGUSR1 nostop noprint
handle SIGUSR2 nostop noprint
handle SIGWAITING nostop noprint
handle SIGLWP nostop noprint
handle SIGPIPE nostop
handle SIGALRM nostop
handle SIGHUP nostop
handle SIGTERM nostop noprint
```

Si vous rencontrez des problémes lors du dèbogage des threads avec gdb, vous devez obtenir la version 5.x de gdb et essayer cela á la place. La nouvelle version de gdb a une meilleur gestion des threads !

Voilá un exemple de comment dèboguer mysqld :

```
shell> gdb /usr/local/libexec/mysqld
gdb> run
...
backtrace full # A faire lorsque mysqld crashe
```

Incluez la sortie suivante dans un mail gènèrè avec `mysqlbug` et envoyez le á `mysql@lists.mysql.com`.

Si `mysqld` ne rèpond plus, vous pouvez utiliser des outils systéme tel que `strace` ou `/usr/proc/bin/pstack` pour savoir oú `mysqld` s'est bloquè.

```
strace /tmp/log libexec/mysqld
```

Si vous utilisez l'interface `DBI` de Perl, vous pouvez activer le dèbogage en utilisant la mèthode `trace` ou en dèfinissant la variable d'environnement `DBI_TRACE`. Voir Section 8.2.2 [Perl DBI Class], page 592.

E.1.4 Utilisation d'un traçage de pile mèmoire

Sur quelques systémes d'exploitation, le log d'erreurs contiendra un fichier de pile mèmoire si `mysqld` se termine soudainement. Vous pouvez utiliser ceci pour trouver oú (et peut-être pourquoi) `mysqld` s'est terminè. Voir Section 4.9.1 [Error log], page 328. Pour obtenir un traçage de la pile, vous ne devez pas compiler `mysqld` avec l'option `-fomit-frame-pointer` de gcc. Voir Section E.1.1 [Compiling for debugging], page 815.

Si le fichier d'erreurs contient quelque chose qui ressemble á ce qui suit :

```
mysqld got signal 11;
The manual section 'Debugging a MySQL server' tells you how to use a
stack trace and/or the core file to produce a readable backtrace that may
help in finding out why mysqld died
Attemping backtrace. You can use the following information to find out
where mysqld died.  If you see no messages after this, something went
terribly wrong
stack range sanity check, ok, backtrace follows
0x40077552
0x81281a0
0x8128f47
```

```
0x8127be0
0x8127995
0x8104947
0x80ff28f
0x810131b
0x80ee4bc
0x80c3c91
0x80c6b43
0x80c1fd9
0x80c1686
```

vous pouvez trouver oú s'est terminè `mysqld` en exècutant ce qui suit :

1. Copiez les nombres prècèdents dans un fichier, '`mysqld.stack`' par exemple.
2. crèez un fichier symbolique pour le serveur `mysqld` :

   ```
   nm -n libexec/mysqld > /tmp/mysqld.sym
   ```

   Notez que beaucoup de distributions binaires MySQL fournissent le fichier prècèdent, nommè `mysqld.sym.gz`. Dans ce cas, dècompressez le en faisant :

   ```
   gunzip < bin/mysqld.sym.gz > /tmp/mysqld.sym
   ```

3. Exècutez `resolve_stack_dump -s /tmp/mysqld.sym -n mysqld.stack`.

   Cela affichera l'endroit oú `mysqld` a plantè. Si cela ne vous aide pas á trouver pourquoi `mysqld` a crashè, vous devez crèer un rapport de bogue et y inclure le rèsultat de la commande prècèdente.

   Notez toutefois que dans la plupart de cas le fait de n'avoir que le traçage de la pile ne nous aidera pas á trouver d'oú vient le probléme. Pour être capable de trouver le bogue, ou fournir une parade, nous aurons besoin dans la plupart des cas, nous aurons besoin de connaître la requête qui a fait planter `mysqld` et une batterie de tests pour que nous puissions reproduire le probléme ! Voir Section 1.6.2.3 [Bug reports], page 27.

E.1.5 Utilisation des fichiers de log pour trouver d'oú viennent les erreurs de mysqld

Notez qu'avant de dèmarrer `mysqld` avec `--log` vous devez vèrifier toutes vos tables avec `myisamchk`. Voir Chapitre 4 [MySQL Database Administration], page 192.

Si `mysqld` se termine ou se bloque, vous devez dèmarrer `mysqld` avec `--log`. Lorsque `mysqld` se termine á nouveau, vous pouvez examiner la fin de votre fichier de log pour trouver les requêtes qui ont terminè `mysqld`.

Si vous utilisez `--log` sans spècifier un nom de fichier, le log est enregistrè dans le dossier des bases de donnèes en tant que 'hostname'.log. Dans la plupart des cas, c'est la derniére requête dans le fichier de log qui a terminè `mysqld`, mais si possible, vèrifiez le en redèmarrant `mysqld` et exècutant á nouveau la requête en question á partir du client en ligne de commande `mysql`. Si elle fonctionne, vous devez aussi tester les autres requêtes complexes qui n'ont pas abouties.

Vous pouvez aussi utiliser la commande `EXPLAIN` sur toutes vos requêtes `SELECT` qui prennent beaucoup de temps á s'exècuter pour être sûrs que `mysqld` utilise les index convenablement. Voir Section 5.2.1 [`EXPLAIN`], page 363.

Vous pouvez trouver les requêtes qui prennent trop de temps á s'exècuter en dèmarrant `mysqld` avec `--log-slow-queries`. Voir Section 4.9.5 [Slow query log], page 331.

Si vous trouvez le texte `mysqld restarted` dans le log d'erreurs (normalement nommè '`hostname.err`') vous avez probablement trouvè une requête qui fait planter `mysqld`. Si tel est le cas, vous devez vèrifier toutes vos tables avec `myisamchk` (voir Chapitre 4 [MySQL Database Administration], page 192), et tester les requêtes dans les fichiers de log MySQL pour voir si elles ne fonctionnent toujours pas. si vous trouvez une requête de ce genre, essayez d'abord de mettre á jour votre version de MySQL en prenant la version la plus rècente. Si cela ne vous aide pas et que vous ne pouvez trouver d'aide dans les archives des mails de `mysql`, vous devez reporter ce bogue á `mysql@lists.mysql.com`. Des liens vers les archives de mails sont disponibles en ligne á l'adresse suivante : `http://lists.mysql.com/`.

Si vous avez dèmarrè `mysqld` avec `myisam-recover`, MySQL vèrifiera et essayera automatiquement de rèparer les tables `MyISAM` si elles sont marquèes comme 'not closed properly' ou 'crashed'. Si cela arrive, MySQL ajoutera une entrèe dans le fichier `hostname.err` '`Warning: Checking table ...`' qui sera suivie de `Warning: Repairing table` si la table devait être rèparèe. si vous obtenez beaucoup de ces erreurs, sans que `mysqld` n'ai crashè juste avant, alors quelque chose ne va pas, et une enquête plus approfondie est nècessaire. Voir Section 4.1.1 [Command-line options], page 192.

Ce n'est bien sûr pas de bon augure si `mysqld` a crashè, mais dans ce cas, il ne faut pas s'attarder sur les messages `Checking table...` mais plutôt essayer de savoir pourquoi `mysqld` a crashè.

E.1.6 Faire une batterie de tests lorsque vous faites face á un probléme de table corrompue

Si vos tables sont corrompues ou que `mysqld` èchoue toujours avec quelques commandes de mises á jour, vous pouvez tester si le bogue est reproduisable en effectuant ce qui suit :

- Coupez le dèmon MySQL (avec `mysqladmin shutdown`).
- Crèez une copie de vos tables (pour prèvoir le cas trés improbable ou la rèparation tournerait mal).
- Vèrifiez toutes les tables avec `myisamchk -s base/*.MYI`. Rèparez toute table corrompue avec `myisamchk -r base/table.MYI`.
- Crèez une seconde copie des tables.
- Effacez (ou dèplacez) tout les vieux fichiers de log du rèpertoire de donnèes de MySQL si vous avez besoin de plus d'espace.
- Dèmarrez `mysqld` avec `--log-bin`. Voir Section 4.9.4 [Binary log], page 329. Si vous voulez trouver une requête qui fait crasher `mysqld`, vous devez utiliser `--log --log-bin`.
- Lorsque vous obtenez une table crashèe, stoppez le `serveur mysqld`.
- Restaurez les sauvegardes.
- Redèmarrez le serveur `mysqld` **sans** `--log-bin`
- Re-exècutez les commandes avec `mysqlbinlog update-log-file | mysql`. Le log des mises á jour est est sauvegardè dans le dossier des donnèes de MySQL avec le nom `hostname-bin.#`.

- Si les tables sont á nouveau corrompues ou que vous pouvez faire èchouer `mysqld` avec la commande prècèdente, vous avez trouvè un bogue reproductible qui devrait être facile á corriger ! Envoyez les tables et le log binaire via FTP á `ftp://support.mysql.com/pub/mysql/secret/` et envoyez un mail á `bugs@lists.mysql.com` ou (si vous êtes client du support) á `support@mysql.com` á propos du probléme et l'èquipe MySQL le corrigera le plus vite possible.

Vous pouvez aussi utiliser le script `mysql_find_rows` pour n'exècuter que quelques requêtes de mises á jour si vous voulez mieux cerner le probléme.

E.2 Dèbogage un client MySQL

Pour pouvoir dèboguer un client MySQL avec le package de dèbogage intègrè, vous devez configurer MySQL avec `--with-debug` ou `--with-debug=full`. Voir Section 2.3.3 [configure options], page 88.

Avant de mettre en marche un client, vous devez dèfinir la variable d'environnement `MYSQL_DEBUG` :

```
shell> MYSQL_DEBUG=d:t:O,/tmp/client.trace
shell> export MYSQL_DEBUG
```

Cela fait gènèrer au client un fichier de traçage dans '`/tmp/client.trace`'.

Si vous avez un probléme avec votre propre code client, vous devez essayer de vous connecter au serveur et exècuter vos requêtes en utilisant un client qui fonctionne. Faites le en utilisant `mysql` en mode dèbogage (en supposant que vous ayez compilè MySQL avec le support du dèbogage) :

```
shell> mysql --debug=d:t:O,/tmp/client.trace
```

Il vous fournira des informations utiles si vous voulez envoyer un rapport de bogue. Voir Section 1.6.2.3 [Bug reports], page 27.

Si votre client crashe au niveau d'un code qui vous parait 'lègal', vous devez vèrifier que votre fichier '`mysql.h`' inclu correspond á votre librairie mysql. Une erreur trés courante est d'utiliser un vieux fichier '`mysql.h`' d'une anciene installation avec la nouvelle librairie MySQL.

E.3 Le package DBUG

Le serveur MySQL et la plupart des clients MySQL sont compilès avec le package DBUG ècrit, á l'origine, par Fred Fish. Lorsque MySQL est compilè avec le support du dèbogage, ce package permet d'obtenir des fichiers de traçage de ce que le programme dèbogue. Voir Section E.1.2 [Making trace files], page 816.

Le package de dèbogage est utilisè en invoquant le programme avec l'option `--debug="..."` ou `-#...`.

La plupart des programmes MySQL ont une chaîne de dèbogage par dèfaut qui sera utilisèe si vous ne spècifiez aucune option á `--debug`. Le fichier de traçage par dèfaut est usuellement `/tmp/nomprogramme.trace` sur Unix et `\nomprogramme.trace` sur Windows.

La chaîne de caractéres de controle du dèbogage est une sèquence de champs sèparès par des deux-points (:) comme celle qui suit :

```
<champ_1>:<champ_2>:...:<champ_N>
```

Chaque champ consiste d'un caractére attribut suivi d'une liste de modificateurs, commençant optionnellement par une virgule, sèparès par des virgules :

```
flag[,modificateur,modificateur,...,modificateur]
```

Les caractéres attributs actuellement reconnus sont :

Attribut	Description
d	Active les sorties des macros DBUG_<N> pour l'état courant. Peut être suivi d'une liste de mots clefs, ce qui sèlectionne la sortie seuelemnt pour les macros DBUG contenant ces mots. Une liste de mots clefs vides implique les sorties de toutes les macros.
D	Attendre aprés chaque ligne rèsultante du dèbogueur. L'arguement est le nombre de dixiémme de secondes á attendre, sujet aux capabilitès de la machine. Et donc, `-#D,20` est une attente de deux secondes.
f	Limiter le dèbogage et/ou le traçage aux fonctions citèes. Notez qu'une liste nulle dèsactivera toutes les fonctions. Les attributs appropriès "d" ou "t" doivent quand même être donnès, cet attribut ne limite que leurs actions si ils sont activès.
F	Identifie le nom du fichier source pour chaque ligne de dèbogage ou de traçage affichèe.
i	Identifie le processus avec son identifiant pour chaque ligne de dèbogage ou de traçage affichèe.
g	Active le profiling. Crèe un fichier nommè 'dbugmon.out' contenant des informations qui peuvent être utilisèes pour profiler le programme. Peut être suivi d'une liste de mots clefs qui sèlectionnent le profiling uniquement pour les fonctions prèsentes dans ctte liste. Une liste nulle implique que toutes les fonctions sont considèrèes.
L	Identifie le numèro de ligne du fichier source pour chaque ligne renvoyèe par le traçage ou le dèbogage.
n	Imprime le niveau de profondeur de la fonction en cours d'exècution pour la sortie du traçage ou du dèbogage.
N	Numèrote chaque ligne de la sortie du dèbogage.
o	Redirige le flux de sortie du dèbogueur vers le fichier spècifiè. Par dèfaut, c'est stderr.
O	Comme `o` mais le fichier est vraiement ècrit entre chaque ajout de contenu. Lorsque le besoin en est, le fichier est fermè puis rèouvert entre chaque ècriture.
p	Limite les actions du dèbogueur aux processus spècifiès. Un processus peut être identifiè avec la macro DBUG_PROCESS et correspondre á un processus dans la liste pour que le dèbogage ait lieu.
P	Affiche le nom du processus courant pour chaque ligne de sortie de dèbogage ou de traçage.
r	Lors du passage á un nouvel ètat, ne pas hèriter le niveau de profondeur de l'ètat de la fonction prècèdente. Utile lorsque l'affichage commence á la marge gauche.
S	Exècute la fonction _sanity(_file_,_line_) sur chaque fonction dèboguèe jusqu'á ce que la valeur de retour de _sanity() différe de 0. (La plupart du temps utilisèe avec safemalloc opur trouver les pertes de mèmoire)
t	Active le traçage des appels/sorties des fonctions. Peut être suivi d'une liste (ne contenant qu'un seul modificateur) donnant un maximum numèrique du traçage, au delá duquel aucune sortie ne sera affichèe pour les macros de dèbogage ou de traçage. Par dèfaut, c'est une option de temps de compilation.

Quelques exemples de chaînes de controle de dèbogage pouvant être utilisèe en ligne de commande dans le shell ("-#" est typiquement utilisè pour introduire une chaîne de controle á un programme d'application) sont :

```
-#d:t
-#d:f,main,subr1:F:L:t,20
-#d,input,output,files:n
-#d:t:i:O,\\mysqld.trace
```

En MySQL, les balises communes affichèes (avec l'option `d`) sont : `enter`,`exit`,`error`,`warning`,`info` et `loop`.

E.4 Mèthodes de verrouillage

Actuellement, MySQL ne supporte que le verrouillage de table pour les tables `ISAM`/`MyISAM` et `HEAP`, le verrouillage de page pour les tables `BDB` et le verrouillage de ligne pour `InnoDB`. Voir Section 5.3.1 [Internal locking], page 380. Avec les tables `MyISAM`, vous pouvez mèlanger librement les `INSERT` et `SELECT` sans poser de verrous (`Versioning`).

Depuis la version 3.23.33, vous pouvez analyser l'utilisation des verrous sur vos tables avec les variables d'environnement `Table_locks_waited` et `Table_locks_immediate`.

Pour dècider si vous voulez utiliser un type de table avec verrouillage de ligne, vous devez commencer par ètudier ce que votre application fait, et quel est le schèma d'utilisation des sèlections et modifications.

Avantages du verrouillage de ligne :

- Moins de conflits de lignes, lorsque les mêmes lignes sont utilisèes par diffèrents threads.
- Moins de modifications pour les annulations (`ROLLBACK`)
- Rend possible le verrouillage d'une ligne pour une longue durèe.

Inconvènients du verrouillage de ligne :

- Prend plus de mèmoire que les verrous de page ou de table.
- Est plus lent que les verrous de page ou de table, lorsqu'il est utilisè sur une grand partie de la table, car il faut alors poser plusieurs verrous.
- Est vraiment bien pire que les autres verrous si vous utilisez souvent la requête `GROUP BY` sur la majeure partie des donnèes, ou si vous avez á scanner toute la table.
- Avec des verrous de plus haut niveau, vous pouvez aussi supporter des verrous d'autres types, pour optimiser l'application, car le coût de l'administration est moindre que pour le verrouillage de ligne.

Les verrous de tables sont supèrieurs aux verrous de page ou de ligne dans les cas suivants :

- Les lectures.
- Les lectures et les modifications sur des clès strictes : c'est le cas si une modification ou un effacement de ligne peut être lu en une seule opèration dans l'index.

  ```
  UPDATE table_name SET column=value WHERE unique_key#
  DELETE FROM table_name WHERE unique_key=#
  ```

- `SELECT` combinè avec `INSERT` (et quelques `UPDATE` et `DELETE` rares).
- De nombreux scans / `GROUP BY` sur toute la table, sans aucune ècriture.

Autres possibiltès alternatives au verrouillage de ligne ou de page :

Le versionnage (comme celui que nous utilisons pour les insertions simultanèes avec MySQL), oú vous pouvez avoir un thread qui ècrit et de nombreux autres qui lisent. Cela signifie que les bases ou tables supportent diffèrentes vues des donnèes, suivants le moment d'accés aux donnèes. D'autres noms pour cette techniques sont `time travel`, `copy on write` ou `copy on demand`.

La `copy on demand` (copie sur demande) est dans de nombreuses situations bien meilleure que le verrouillage de page ou de ligne. Le pire reste l'utilisation de mèmoire, qui est bien plus forte qu'avec les verrous normaux.

Au lieu d'utiliser le verrouillage de ligne, vous pouvez utiliser des verrous au niveau de l'application (comme les get_lock/release_lock de MySQL). Cela ne fonctionne qu'avec les applications bien èlevèes.

Dans de nombreux cas, vous pouvez faire une estimation èclairèe du type de verrouillage qui est le meilleur pour votre application, mais la plus part du temps, il est trés difficile de dire si un type de verrouillage est plus adaptè que l'autre. Tout dèpend de votre application, et des diffèrentes parties qui rèclament des verrous.

Voici quelques conseils sur le verrouillage de MySQL :

La plupart des sites web fond de nombreuses lecture, trés peu d'effacements, des modifications avec des clès, et des insertions dans des tables spècifiques. La base MySQL est trés bien optimisèe pour cela.

Les utilisateurs simultanès ne sont pas un probléme si ils ne mèlangent pas les insertions et les modifications, qui dèclenchent l'analyse de nombreuses lignes dans la même table.

Si vous mèlangez les insertions et les effacements sur les mêmes tables, alors utilisez `INSERT DELAYED` pour amèliorer la situation.

Vous pouvez aussi utiliser la commande `LOCK TABLES` pour accèlèrer les opèrations (de nombreuses modifications dans le même verrou et bien plus rapide que de le faire sans le verrou). Rèpartir les opèrations sur plusieurs tables peut aussi aider.

Si vous avez des problémes avec les verrous de tables de MySQL, vous pourriez les rèsoudre en convertissant les tables au format `InnoDB` ou `BDB` . Voir Section 7.5 [InnoDB], page 544. Voir Section 7.6 [BDB], page 585.

Le chapitre d'optimisation du manuel couvre de nombreux aspects du paramètrage de l'application. Voir Section 5.2.12 [Tips], page 377.

E.5 Commentaires á propos des threads RTS

J'ai essayè d'utiliser le package de threads RTS avec MySQL mais je suis restè bloquè au niveau des problémes suivants :

Ils utilisent une veille version avec beaucoup d'appels POSIX et il est vraiment difficile de crèer une couche d'abstraction pour toutes les fonctions. Je pense qu'il serait plus facile de changer la librairie des threads pour qu'elle suive les nouvelles spècifications POSIX.

Quelques couches d'abstractions sont dèjá ècrites. Voyez '`mysys/my_pthread.c`' pour plus d'informations.

Au minimum, ce qui suit devra être changè :

`pthread_get_specific` doit utiliser un seul argument. `sigwait` doit prendre deux arguments. Beaucoup de fonctions (du moins `pthread_cond_wait`, `pthread_cond_timedwait`) doivent retourner le code erreur lorsqu'elles en rencontrent. Elle retournent á prèsent -1 et dèfinissent `errno`.

Un autre probléme est que les threads au niveau utilisateurs utilisent les signaux `ALRM` et que ceux-ci fait èchouer beaucoup de fonctions (`read`, `write`, `open`...). MySQL devrait faire une autre tentative á chaque interruption mais cela n'est pas facile á vèrifier.

Le plus gros probléme non-rèsolu est le suivant :

Pour avoir des alertes au niveau des threads, j'ai changè '`mysys/thr_alarm.c`' pour avoir une attente entre les alarmes avec `pthread_cond_timedwait()`, mais cela èchoue avec une erreur `EINTR`. J'ai essayè de dèboguer la librairie des threads pour voir d'oú cela venait, mais je n'ai pu trouver aucune solution simple.

Si quelqu'un veut utiliser MySQL avec les threads RTS je suggére ce qui suit :

- Changez les fonctions que MySQL utilise á partir de la librairie des threads en POSIX. Cela ne devrait pas vous prendre beaucoup de temps.
- Compilez toutes les librairies avec `-DHAVE_rts_threads`.
- Compilez `thr_alarm`.
- S'il y'a de petites diffèrences dans l'implèmentation, elles peuvent être corrigèes en changeant les fichiers '`my_pthread.h`' et '`my_pthread.c`'.
- Exècutez `thr_alarm`. S'il tourne sans aucun message du type "warning", "error" ou "aborted", vous êtes sur le bon chemin. Voici une bonne exècution se dèroulant sur Solaris :

```
Main thread: 1
Thread 0 (5) started
Thread: 5  Waiting
process_alarm
Thread 1 (6) started
Thread: 6  Waiting
process_alarm
process_alarm
thread_alarm
Thread: 6  Slept for 1 (1) sec
Thread: 6  Waiting
process_alarm
process_alarm
thread_alarm
Thread: 6  Slept for 2 (2) sec
Thread: 6  Simulation of no alarm needed
Thread: 6  Slept for 0 (3) sec
Thread: 6  Waiting
process_alarm
process_alarm
thread_alarm
Thread: 6  Slept for 4 (4) sec
Thread: 6  Waiting
```

```
process_alarm
thread_alarm
Thread: 5  Slept for 10 (10) sec
Thread: 5  Waiting
process_alarm
process_alarm
thread_alarm
Thread: 6  Slept for 5 (5) sec
Thread: 6  Waiting
process_alarm
process_alarm

...
thread_alarm
Thread: 5  Slept for 0 (1) sec
end
```

E.6 Diffèrences entre les diffèrents packages de threads

MySQL est trés dèpendant du package de threads utilisè. Ce qui fait que lors du choix d'une bonne plate-forme pour MySQL, le package des threads est trés important.

Il y'a au moins trois types de packages de threads :

- Threads utilisateurs dans un processus unique. Le changement de threads est gèrè avec des alarmes et la librairie des threads gére les fonctions non-utilisables avec les threads par des verrouillages. Les opèrations de lectures, d'ècritures et de sèlections sont usuellement gèrèes avec une sèlection spècifique aux threads qui passe á un autre thread si celui en cours d'exècution attend des donnèes. Si les packages des threads utilisateurs sont intègrès dans les librairies standards (threads FreeBSD et BSDI) le package de thread nècessite moins de ressources que les packages de thread qui doivent mapper toutes les appels non-sûrs (MIT-pthreads, FSU Pthreads et RTS threads). Avec quelques environnements, (SCO par exemple), tous les appels systéme sont sûrs pour les threads, ce qui fait que la liaison peut se faire trés facilement (Pthreads FSU sur SCO). Mauvais côtè : Tous les appels mappès prennent un peu de temps, et il est assez difficile de pouvoir gèrer toutes les situations. Il y'a aussi souvent des appels systéme qui ne sont pas gèrès par le package de threads (comme MIT-pthreads et les sockets). La gestion des threads n'est pas toujours optimale.
- Threads utilisateurs dans des processus sèparès. Les changements de threads sont effectuès par le noyau et toutes les donnèes sont partagèes entre les threads. Le package de thread gére les appels threads standards pour permettre le partage entre les threads. LinuxThreads utilise cette mèthode. Mauvais côtè : Beaucoup de processus. La crèation des threads est lente. Si un thread s'interrompt, les autres restent en suspens et vous devez tous les terminer avant de redèmarrer. Le changement de thread est d'une certaine façon consommateur de ressources.
- Threads noyau. Le changement de threads est gèrè par la librairie de threads ou le noyau est trés rapide. Tout est fait en un seul processus, mais sur certains systémes,

`ps` peut montrer plusieurs threads. Si un thread èchoue, tout le processus èchoue. La plupart des appels systéme sont bons pour les threads et ne devraient avoir besoin que d'une petite perte de performances. Solaris, HP-UX, AIX et OSF/1 ont des threads noyau.

Avec quelques systémes, les threads du noyau sont gèrès en intègrant les threads niveau utilisateur dans les librairies du systéme. Dans ces cas, le changement de thread ne peut être fait qu'avec la librairie des threads et le noyau n'est pas vraiment "attentif aux threads".

Annexe F Variables d'environnement

Voici une liste de toutes les variables d'environnement qui sont directement ou indirectement utilisèes par MySQL. La plupart peuvent être retrouvèes dans d'autres parties du manuel.

Notez que toute option en ligne de commande prendra la prècèdence par rapport aux valeurs spècifièes dans les fichiers de configurations et les variables d'environnement, et que les valeurs dans les fichiers de configurations prennent la prècèdence sur les valeurs des variables d'environnement.

Dans beaucoup de cas, il est prèfèrable d'utiliser un fichier de configuration plutôt que les variables d'environnement pour modifier le comportement de MySQL. Voir Section 4.1.2 [Option files], page 198.

Variable	Description
`CCX`	A passer au compilateur C++ lors de l'exècution de configure.
`CC`	A passer au compilateur C lors de l'exècution de configure.
`CFLAGS`	A passer au compilateur C lors de l'exècution de configure.
`CXXFLAGS`	A passer au compilateur C++ lors de l'exècution de configure.
`DBI_USER`	L'utilisateur par dèfaut pour Perl DBI.
`DBI_TRACE`	Utilisèe lors du traçage de Perl DBI.
`HOME`	Le chemin par dèfaut vers les fichiers d'hostorique de `mysql` est '`$HOME/.mysql_history`'.
`LD_RUN_PATH`	Utilisè pour spècifier oú se trouve '`libmysqlclient.so`'.
`MYSQL_DEBUG`	Options de dèbogage/traçage lors de la phase de dèbogage.
`MYSQL_HISTFILE`	Le chemin vers le fichier d'historique de `mysql`.
`MYSQL_HOST`	Nom d'hôte par dèfaut utilisè par le client en ligne de commande `mysql`.
`MYSQL_PS1`	Invite de commande á utiliser dans le client en ligne de commande `mysql`. Voir Section 4.8.2 [mysql], page 307.
`MYSQL_PWD`	Le mot de passe par dèfaut lors de la connexion á `mysqld`. Notez que cette utilisation n'est pas sècurisèe.
`MYSQL_TCP_PORT`	Le port TCP/IP par dèfaut.
`MYSQL_UNIX_PORT`	La socket par dèfaut; utilisèe pour les connexion á `localhost`.
`PATH`	Utilisè par le shell pour trouver les programmes MySQL.
`TMPDIR`	Le dossier ou les tables et fichiers temporaires sont crèes.
`TZ`	Cela devrait être configurè á votre fuseau horaire local. Voir Section A.4.6 [Timezone problems], page 702.
`UMASK_DIR`	Le masque (mask) de crèation des dossiers. Notez que cette valeur est soumise á un ET logique par rapport `UMASK` !
`UMASK`	Le masque (mask) de crèation des fichiers utilisateurs lors de la crèation de fichiers.
`USER`	L'utilisateur par dèfaut de Windows á utiliser lors de la connexion á `mysqld`.

Annexe G Expressions règuliéres MySQL

Une expression règuliére (regex) est la meilleure mèthode pour spècifier une recherche complexe.

MySQL utilise l'implèmentation de Henry Spencer des expressions règuliéres qui tend á être conforme á POSIX 1003.2. MySQL en utilise la version ètendue.

Ceci est une rèfèrence simplifiée qui n'aborde pas les dètails. Pour avoir plus d'ionformations dètaillées, regardez la page de manuel `regex(7)` de Henry Spencer qui est inclue dans la distribution des sources. Voir Annexe C [Credits], page 717.

Une expression règuliére dècrit un jeu de chaînes de caractéres. La plus simple est celle qui ne comporte pas de caractéres spèciaux. Par exemple, la regexp `bonjour` trouvera `bonjour` et rien d'autre.

Les expression règuliéres non-trivialles utilisent des constructions spèciales pour pouvoir trouver plus d'une chaîne. Par exemple, la regexp `bonjour|monde` trouve la chaîne `bonjour` ou la chaîne `monde`.

Voici un exemple encore plus complexe : la regexp `B[an]*s` trouve l'une des chaînes suivantes `Bananas`, `Baaaaas`, `Bs`, et n'importe quelle autre chaîne commençant par un `B`, se terminant par un `s`, et contenant n'importe quel nombre de `a` et de `n` au milieu.

Une expression règuliére peut utiliser l'un des caractéres spèciaux ou constructions suivants :

`^` Correspond au dèbut de la chaîne.

```
mysql> SELECT "fo\nfo" REGEXP "^fo$";          -> 0
mysql> SELECT "fofo" REGEXP "^fo";             -> 1
```

`$` Correspond á la fin de la chaîne.

```
mysql> SELECT "fo\no" REGEXP "^fo\no$";        -> 1
mysql> SELECT "fo\no" REGEXP "^fo$";           -> 0
```

`.` N'importe quel caractére (nouvelle ligne inclus).

```
mysql> SELECT "fofo" REGEXP "^f.*";            -> 1
mysql> SELECT "fo\nfo" REGEXP "^f.*";          -> 1
```

`a*` Correspond á toute sèquence de zèro ou plus caractéres `a`.

```
mysql> SELECT "Ban" REGEXP "^Ba*n";            -> 1
mysql> SELECT "Baaan" REGEXP "^Ba*n";          -> 1
mysql> SELECT "Bn" REGEXP "^Ba*n";             -> 1
```

`a+` Correspond á toute sèquence de un ou plus caractéres `a`.

```
mysql> SELECT "Ban" REGEXP "^Ba+n";            -> 1
mysql> SELECT "Bn" REGEXP "^Ba+n";             -> 0
```

`a?` Correspond á zèro ou un caractére `a`.

```
mysql> SELECT "Bn" REGEXP "^Ba?n";             -> 1
mysql> SELECT "Ban" REGEXP "^Ba?n";            -> 1
mysql> SELECT "Baan" REGEXP "^Ba?n";           -> 0
```

`de|abc` Correspond aux sèquences de `de` ou de `abc`.

```
mysql> SELECT "pi" REGEXP "pi|apa";           -> 1
mysql> SELECT "axe" REGEXP "pi|apa";          -> 0
mysql> SELECT "apa" REGEXP "pi|apa";          -> 1
mysql> SELECT "apa" REGEXP "^(pi|apa)$";      -> 1
mysql> SELECT "pi" REGEXP "^(pi|apa)$";       -> 1
mysql> SELECT "pix" REGEXP "^(pi|apa)$";      -> 0
```

`(abc)*` Correspond á zèro ou plus sèquences de `abc`.

```
mysql> SELECT "pi" REGEXP "^(pi)*$";          -> 1
mysql> SELECT "pip" REGEXP "^(pi)*$";         -> 0
mysql> SELECT "pipi" REGEXP "^(pi)*$";        -> 1
```

`{1}`
`{2,3}` Voici une façon plus gènèrale d'ècrire les regexps qui correspondent á plusieurs occurences du dernier atome.

`a*` Peut être ècrit `a{0,}`.

`a+` Peut être ècrit `a{1,}`.

`a?` Peut être ècrit `a{0,1}`.

Pour être plus prècis, un atome suivi d'une accolade contenant un entier `i` et pas de virgule trouve une sèquence de exactement `i` atomes.

Un atome suivi d'une accolade contenant un entier `i` et une virgule trouve une sèquence de `i` ou plus atomes.

Un atome suivi d'une accolade contenant deux entiers `i` et `j` sèparès d'une virgule trouve les sèquences de `i` á `j` (inclusif) atomes.

Les deux arguements doivent être compris entre `0` et `RE_DUP_MAX` (par dèfaut 255), inclusif. S'il y'a deux arguements, le second doit être supèrieur ou ègal au premier.

`[a-dX]`
`[^a-dX]` Trouve n'importe quel caractére qui est (ou n'est pas, si `^` est utilisè) `a`, `b`, `c`, `d` ou `X`. Pour inclure le caractére litèral `]`, il doit suivre immèdiatement le crochet ouvrant `[`. Pour inclure le caractére litèral `-`, il doit être ècrit en premier ou en dernier. Ce qui fait que `[0-9]` correspond á n'importe quel chiffre. Chaque caractére qui n'a pas de signification spèciale á l'intèrieur une paire de `[]` ne joue pas de rôle spècial et ne correspond qu'á lui même.

```
mysql> SELECT "aXbc" REGEXP "[a-dXYZ]";       -> 1
mysql> SELECT "aXbc" REGEXP "^[a-dXYZ]$";     -> 0
mysql> SELECT "aXbc" REGEXP "^[a-dXYZ]+$";    -> 1
mysql> SELECT "aXbc" REGEXP "^[^a-dXYZ]+$";   -> 0
mysql> SELECT "gheis" REGEXP "^[^a-dXYZ]+$";  -> 1
mysql> SELECT "gheisa" REGEXP "^[^a-dXYZ]+$"; -> 0
```

`[[.charactéres.]]`
La sèquence de caractéres de cet èlèment d'assemblage. La sèquence est un èlèment de la liste contenue entre les crochets. Une telle expression contenant un èlèment d'assemblage multi-caractéres peut ainsi trouver plus d'un caractére.

Par exemple, si la sèquence d'assemblage inclut un èlèment `ch`, alors l'expression reguliére `[[.ch.]]*c` trouve les cinq premiers caractéres de `chchcc`.

`[=character_class=]`

Une classe d'èquivalence, remplaçant les sèquences de caractéres de tous les èlèments de l'assemblage èquivalents á celui-ci, lui même inclut.

Par exemple, si `o` et `(+)` sont membres d'une classe d'èquivalence, alors `[[=o=]]`, `[[=(+)=]]`, et `[o(+)]` sont tous des synonymes. Une classe d'èquivalence ne doit pas être un point final d'intervalle.

`[:classe_de_caractéres:]`

Dans une expression entre crochets, le nom d'une classe de caractéres entourèe de `[:` et `:]` remplace la liste de tous les caractéres appartenant á cette classe. Les noms des classes de caractéres sont :

Nom	**Nom**	**Nom**
alnum	digit	punct
alpha	graph	space
blank	lower	upper
cntrl	print	xdigit

Voilá les classes de caractéres dèfinies dans la page de manuel `ctype(3)`. Une locale peut en fournir d'autres. Une classe de caractére ne doit pas être utilisèe en tant que point final d'intervalle.

```
mysql> SELECT "justalnums" REGEXP "[[:alnum:]]+";       -> 1
mysql> SELECT "!!" REGEXP "[[:alnum:]]+";               -> 0
```

`[[:<:]]`

`[[:>:]]` Ceux lá trouvent la chaîne nulle qui prècéde et suit chaque mot. Un mot est dèfini comme ètant une sèquence de caractéres qui n'est ni suivi ni prècèdèe d'un caractére de mot. Un caractére de mot est un caractére alnum (dèfini par `ctype(3)`) ou un tiret bas (_).

```
mysql> SELECT "a word a" REGEXP "[[:<:]]word[[:>:]]";      -> 1
mysql> SELECT "a xword a" REGEXP "[[:<:]]word[[:>:]]";     -> 0
```

```
mysql> SELECT "weeknights" REGEXP "^(wee|week)(knights|nights)$"; -> 1
```

Annexe H Licence Publique Gènèrale GNU

Notice d'accompagnement de la traduction non officielle á conserver dans toute reproduction de cette traduction

This is an unofficial translation of the GNU General Public License into french. It was not published by the Free Software Foundation, and does not legally state the distribution terms for software that uses the GNU GPL–only the original English text of the GNU GPL does that. However, we hope that this translation will help french speakers understand the GNU GPL better.

Ceci est une traduction non officielle de la GNU General Public License en français. Elle n'a pas ètè publiée par la Free Software Foundation, et ne dètermine pas les termes de distribution pour les logiciels qui utilisent la GNU GPL–seul le texte anglais original de la GNU GPL en a le droit. Cependant, nous espèrons que cette traduction aidera les francophones á mieux comprendre la GPL.

Cette traduction est sous Copyright 2001 APRIL (`http://www.april.org`). La version la plus á jour de ce document est disponible sur `http://www.april.org/gnu/gpl_french.html`

Il est permis á tout le monde de reproduire et distribuer des copies conformes de cette traduction, mais aucune modification ne doit y être apportèe, et la prèsente notice doit être prèservèe.

Nous autorisons la FSF á apporter toute modification qu'elle jugera nècessaire pour rendre la traduction plus claire.

Version 2, June 1991

Il est permis á tout le monde de reproduire et distribuer des copies conformes de ce document de licence, mais aucune modification ne doit y être apportèe.

H.1 Prèambule

Les licences relatives á la plupart des logiciels sont destinèes á supprimer votre libertè de les partager et de les modifier. Par contraste, la licence publique gènèrale GNU General Public License veut garantir votre libertè de partager et de modifier les logiciels libres, pour qu'ils soient vraiment libres pour tous leurs utilisateurs. La prèsente licence publique gènèrale s'applique á la plupart des logiciels de la Free Software Foundation, ainsi qu'á tout autre programme dont les auteurs s'engagent á l'utiliser. (Certains autres logiciels sont couverts par la Licence Publique Gènèrale pour Bibliothéques GNU á la place). Vous pouvez aussi l'appliquer á vos programmes.

Quand nous parlons de logiciels libres, nous parlons de libertè, non de gratuitè. Nos licences publiques gènèrales veulent vous garantir que vous avez toute libertè de distribuer des copies des logiciels libres (et de facturer ce service, si vous le souhaitez), que vous recevez les codes sources ou pouvez les obtenir si vous le souhaitez, que vous pouvez modifier les logiciels ou en utiliser des èlèments dans de nouveaux programmes libres, et que vous savez que vous pouvez le faire.

Pour protèger vos droits, nous devons apporter des restrictions, qui vont interdire á quiconque de vous dènier ces droits, ou de vous demander de vous en dèsister. Ces restrictions se traduisent par certaines responsabilitès pour ce qui vous concerne, si vous distribuez des copies de logiciels, ou si vous les modifiez.

Par exemple, si vous distribuez des copies d'un tel programme, gratuitement ou contre une rèmunèration, vous devez transfèrer aux destinataires tous les droits dont vous disposez. Vous devez vous garantir qu'eux-mêmes, par ailleurs, reçoivent ou peuvent recevoir le code source. Et vous devez leur montrer les prèsentes dispositions, de façon qu'ils connaissent leurs droits.

Nous protègeons vos droits en deux ètapes : (1) Nous assurons le droit d'auteur (copyright) du logiciel, et (2) Nous vous proposons cette licence, qui vous donne l'autorisation lègale de dupliquer, distribuer et/ou modifier le logiciel.

De même, pour la protection de chacun des auteurs, et pour notre propre protection, nous souhaitons nous assurer que tout le monde comprenne qu'il n'y a aucune garantie portant sur ce logiciel libre. Si le logiciel est modifiè par quelqu'un d'autre puis transmis á des tiers, nous souhaitons que les destinataires sachent que ce qu'ils possédent n'est pas l'original, de façon que tous problémes introduits par d'autres ne se traduisent pas par une rèpercussion nègative sur la rèputation de l'auteur original.

Enfin, tout programme libre est en permanence menacè par des brevets de logiciels. Nous souhaitons èviter le danger que des sous-distributeurs d'un programme libre obtiennent á titre individuel des licences de brevets, avec comme consèquence qu'ils ont un droit de propriètè sur le programme. Pour èviter cette situation, nous avons fait tout ce qui est nècessaire pour que tous brevets doivent faire l'objet d'une concession de licence qui en permette l'utilisation libre par quiconque, ou bien qu'il ne soit pas concèdè du tout.

Nous prèsentons ci-dessous les clauses et dispositions concernant la duplication, la distribution et la modification.

H.2 TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. Le prèsent contrat de licence s'applique á tout programme ou autre ouvrage contenant un avis, apposè par le dètenteur du droit de propriètè, disant qu'il peut être distribuè au titre des dispositions de la prèsente Licence Publique Gènèrale. Ci-aprés, le "Programme" dèsigne l'un quelconque de ces programmes ou ouvrages, et un "ouvrage fondè sur le programme" dèsigne soit le programme, soit un ouvrage qui en dèrive au titre de la loi sur le droit d'auteur ; plus prècisèment, il s'agira d'un ouvrage contenant le programme ou une version de ce dernier, soit mot á mot, soit avec des modifications et/ou traduit en une autre langue (ci-aprés, le terme "modification" englobe, sans aucune limitation, les traductions qui en sont faites). Chaque titulaire de licence sera appelè "concessionnaire".

   Les activitès autres que la duplication, la distribution et la modification ne sont pas couvertes par la prèsente licence ; elles n'entrent pas dans le cadre de cette derniére. L'exècution du programme n'est soumise á aucune restriction, et les rèsultats du programme ne sont couverts que si son contenu constitue un ouvrage fondè sur le programme (indèpendamment du fait qu'il a ètè rèalisè par exècution du programme). La vèracitè de ce qui prècéde dèpend de ce que fait le programme.

1. Le concessionnaire peut dupliquer et distribuer des copies mot á mot du code source du programme tel qu'il les reçoit, et ce sur un support quelconque, du moment qu'il appose, d'une maniére parfaitement visible et approprièe, sur chaque exemplaire, un avis appropriè de droits d'auteur (Copyright) et de renonciation á garantie ; qu'il maintient intacts tous les avis qui se rapportent á la prèsente licence et á l'absence de toute garantie ; et qu'il transmet á tout destinataire du programme un exemplaire de la prèsente licence en même temps que le programme.

   Le concessionnaire peut facturer l'acte physique de transfert d'un exemplaire, et il peut, á sa discrètion, proposer en èchange d'une rèmunèration une protection en garantie.

2. Le concessionnaire peut modifier son ou ses exemplaires du programme ou de toute portion de ce dernier, en formant ainsi un ouvrage fondè sur le programme, et dupliquer et distribuer ces modifications ou cet ouvrage selon les dispositions de la section 1 ci-dessus, du moment que le concessionnaire satisfait aussi á toutes ces conditions :

   a. Le concessionnaire doit faire en sorte que les fichiers modifiès portent un avis, parfaitement visible, disant que le concessionnaire a modifiè les fichiers, avec la date de tout changement.

   b. Le concessionnaire doit faire en sorte que tout ouvrage qu'il distribue ou publie, et qui, en totalitè ou en partie, contient le programme ou une partie quelconque de ce dernier ou en dèrive, soit concèdè en bloc, á titre gracieux, á tous tiers au titre des dispositions de la prèsente licence.

   c. Si le programme modifiè lit normalement des instructions interactives lors de son exècution, le concessionnaire doit, quand il commence l'exècution du programme pour une telle utilisation interactive de la maniére la plus usuelle, faire en sorte que ce programme imprime ou affiche une annonce, comprenant un avis appropriè de droits d'auteur, et un avis selon lequel il n'y a aucune garantie (ou autrement, que le concessionnaire fournit une garantie), et que les utilisateurs peuvent redistribuer le programme au titre de ces dispositions, et disant á l'utilisateur comment visualiser une copie de cette licence (exception : si le programme par lui-même est interactif mais n'imprime normalement pas une telle annonce, l'ouvrage du concessionnaire se fondant sur le programme n'a pas besoin d'imprimer une annonce).

   Les exigences ci-dessus s'appliquent á l'ouvrage modifiè pris en bloc. Si des sections identifiables de cet ouvrage ne dèrivent pas du programme et peuvent être considèrèes raisonnablement comme reprèsentant des ouvrages indèpendants et distincts par eux-mêmes, alors la prèsente licence, et ses dispositions, ne s'appliquent pas á ces sections quand le concessionnaire les distribue sous forme d'ouvrages distincts. Mais quand le concessionnaire distribue ces mêmes sections en tant qu'èlèment d'un tout qui reprèsente un ouvrage se fondant sur le programme, la distribution de ce tout doit se faire conformèment aux dispositions de la prèsente licence, dont les autorisations, portant sur d'autres concessionnaires, s'ètendent á la totalitè dont il est question, et ainsi á chacune de ces parties, indèpendamment de celui qu'il a ècrite.

   Ainsi, cette section n'a pas pour but de revendiquer des droits ou de contester vos droits sur un ouvrage entiérement ècrit par le concessionnaire ; bien plus, l'intention est d'exercer le droit de surveiller la distribution d'ouvrages dèrivèe ou collective se fondant sur le programme.

De plus, un simple assemblage d'un autre ouvrage ne se fondant pas sur le programme, avec le programme (ou avec un ouvrage se fondant sur le programme) sur un volume d'un support de stockage ou distribution, ne fait pas entrer l'autre ouvrage dans le cadre de la prèsente licence.

3. Le concessionnaire peut dupliquer et distribuer le programme (ou un ouvrage se fondant sur ce dernier, au titre de la Section 2), en code objet ou sous une forme exècutable, au titre des dispositions des Sections 1 et 2 ci-dessus, du moment que le concessionnaire effectue aussi l'une des opèrations suivantes :
   a. Lui joindre le code source complet correspondant, exploitable par une machine, code qui doit être distribuè au titre des Sections 1 et 2 ci-dessus sur un support couramment utilisè pour l'èchange de logiciels ; ou bien
   b. Lui joindre une offre ècrite, dont la validitè se prolonge pendant au moins 3 ans, de transmettre á un tiers quelconque, pour un montant non supèrieur au coût pour le concessionnaire, de rèalisation physique de la distribution de la source, un exemplaire complet, exploitable par une machine, du code source correspondant, qui devra être distribuè au titre des dispositions des Sections 1 et 2 ci-dessus sur un support couramment utilisè pour l'èchange des logiciels ; ou bien
   c. Lui joindre les informations que le concessionnaire a reçues, pour proposer une distribution du code source correspondant (cette variante n'est autorisèe que pour la distribution non commerciale, et seulement si le concessionnaire a reçu le programme sous forme exècutable ou sous forme d'un code objet, avec une telle offre, conformèment á l'alinèa b) ci-dessus).

   Le code source d'un ouvrage reprèsente la forme prèfèrèe de l'ouvrage pour y effectuer des modifications. Pour un ouvrage exècutable, le code source complet reprèsente la totalitè du code source pour tous les modules qu'il contient, plus tous fichiers de dèfinitions d'interface associès, plus les informations en code machine pour commander la compilation et l'installation du programme exècutable. Cependant, á titre d'exceptions spèciales, le code source distribuè n'a pas besoin de comprendre quoi que ce soit qui est normalement distribuè (sous forme source ou sous forme binaire) avec les composants principaux (compilateur, noyau de systéme d'exploitation, etc.) du systéme d'exploitation sur lequel est exècutè le programme exècutable, á moins que le composant, par lui-même, soit joint au programme exècutable.

   Si la distribution de l'exècutable ou du code objet est rèalisèe de telle sorte qu'elle offre d'accèder á une copie á partir d'un lieu dèsignè, alors le fait d'offrir un accés èquivalent á la duplication du code source á partir de ce même lieu s'entend comme distribution du code source, même si des tiers ne sont pas contraints de dupliquer la source en même temps que le code objet.

4. Le concessionnaire ne peut dupliquer, modifier, concèder en sous-licence ou distribuer le programme, sauf si cela est expressèment prèvu par les dispositions de la prèsente licence. Toute tentative pour autrement dupliquer, modifier, concèder en sous-licence ou distribuer le programme est rèpètèe nulle, et met automatiquement fin aux droits du concessionnaire au titre de la prèsente licence. Cependant, les parties qui ont reçu des copies, ou des droits, de la part du concessionnaire au titre de la prèsente licence, ne verront pas expirer leur contrat de licence, tant que ces parties agissent d'une maniére parfaitement conforme.

5. Il n'est pas exigè du concessionnaire qu'il accepte la prèsente licence, car il ne l'a pas signèe. Cependant, rien d'autre n'octroie au concessionnaire l'autorisation de modifier ou de distribuer le programme ou ses ouvrages dèrivès. Ces actions sont interdites par la loi si le concessionnaire n'accepte pas la prèsente licence. En consèquence, par le fait de modifier ou de distribuer le programme (ou un ouvrage quelconque se fondant sur le programme), le concessionnaire indique qu'il accepte la prèsente licence, et qu'il a la volontè de se conformer á toutes les clauses et dispositions concernant la duplication, la distribution ou la modification du programme ou d'ouvrages se fondant sur ce dernier.
6. Chaque fois que le concessionnaire redistribue le programme (ou tout ouvrage se fondant sur le programme), le destinataire reçoit automatiquement une licence de l'èmetteur initial de la licence, pour dupliquer, distribuer ou modifier le programme, sous rèserve des prèsentes clauses et dispositions. Le concessionnaire ne peut imposer aucune restriction plus poussèe sur l'exercice, par le destinataire, des droits octroyès au titre des prèsentes. Le concessionnaire n'a pas pour responsabilitè d'exiger que des tiers se conforment á la prèsente licence.
7. Si, en consèquence une dècision de justice ou une allègation d'infraction au droit des brevets, ou pour toute autre raison (qui n'est pas limitèe á des problémes de propriètès industrielles), des conditions sont imposèes au concessionnaire (par autoritè de justice, par convention ou autrement), qui entrent en contradiction avec les dispositions de la prèsente licence, elles n'exemptent pas le concessionnaire de respecter les dispositions de la prèsente licence. Si le concessionnaire ne peut procèder á la distribution de façon á satisfaire simultanèment á ces obligations au titre de la prèsente licence et á toutes autres obligations pertinentes, alors, en consèquence de ce qui prècéde, le concessionnaire peut ne pas procèder du tout á la distribution du programme. Par exemple, si une licence de brevet ne permettait pas une redistribution du programme, sans redevances, par tous ceux qui reçoivent des copies directement ou indirectement par l'intermèdiaire du concessionnaire, alors le seul moyen par lequel le concessionnaire pourrait satisfaire tant á cette licence de brevet qu'á la prèsente licence, consisterait á s'abstenir complétement de distribuer le programme.

   Si une partie quelconque de cette section est considèrèe comme nulle ou non exècutoire dans certaines circonstances particuliéres, le reste de cette section est rèputè s'appliquer, et la section dans son ensemble est considèrèe comme s'appliquant dans les autres circonstances.

   La prèsente section n'a pas pour objet de pousser le concessionnaire á enfreindre tous brevets ou autres revendications á droit de propriètè, ou encore á contester la validitè de une ou plusieurs quelconques de ces revendications ; la prèsente section a pour objet unique de protèger l'intègritè du systéme de distribution des logiciels libres, systéme qui est mis en oeuvre par les pratiques liées aux licences publiques. De nombreuses personnes ont apportè une forte contribution á la gamme ètendue des logiciels distribuès par ce systéme, en comptant sur l'application systèmatique de ce systéme ; c'est á l'auteur/donateur de dècider s'il a la volontè de distribuer le logiciel par un quelconque autre systéme, et un concessionnaire ne peut imposer ce choix.

   La prèsente section veut rendre parfaitement claire ce que l'on pense être une consèquence du reste de la prèsente licence.
8. Si la distribution et/ou l'utilisation du Programme est restreinte dans certains pays, sous l'effet de brevets ou d'interfaces prèsentant un droit d'auteur, le dètenteur du droit

d'auteur original, qui soumet le Programme aux dispositions de la prèsente licence, pourra ajouter une limitation expresse de distribution gèographique excluant ces pays, de façon que la distribution ne soit autorisèe que dans les pays ou parmi les pays qui ne sont pas ainsi exclus. Dans ce cas, la limitation fait partie intègrante de la prèsente licence, comme si elle ètait ècrite dans le corps de la prèsente licence.

9. La Free Software Foundation peut, de temps á autre, publier des versions rèvisèes et/ou nouvelles du General Public License. Ces nouvelles versions seront analogues, du point de vue de leur esprit, á la prèsente version, mais pourront en diffèrer dans le dètail, pour rèsoudre de nouveaux problémes ou de nouvelles situations.

   Chaque version reçoit un numèro de version qui lui est propre. Si le programme spècifie un numèro de version de la prèsente licence, qui s'applique á cette dernier et "á toute autre version ultèrieure", le concessionnaire a le choix de respecter les clauses et dispositions de cette version, ou une quelconque version ultèrieure publièe par la Free Software Foundation. Si le programme ne spècifie pas de numèro de version de la prèsente licence, le concessionnaire pourra choisir une version quelconque publièe á tout moment par la Free Software Foundation.

10. Si le concessionnaire souhaite incorporer des parties du programme dans d'autres programmes libres dont les conditions de distribution sont diffèrentes, il devrait ècrire á l'auteur pour demander son autorisation. Pour un logiciel soumis á droit d'auteur par la Free Software Foundation, il devra ècrire á la Free Software Foundation ; nous faisons quelquefois des exceptions á cette régle. Notre dècision va être guidèe par le double objectif de protèger le statut libre de tous les dèrivès de nos logiciels libres, et de favoriser le partage et la rèutilisation des logiciels en gènèral.

NO WARRANTY

11. COMME LA LICENCE DU PROGRAMME EST CONCEDEE A TITRE GRATUIT, IL N'Y AUCUNE GARANTIE S'APPLIQUANT AU PROGRAMME, DANS LA MESURE AUTORISEE PAR LA LOI EN VIGUEUR. SAUF MENTION CONTRAIRE ECRITE, LES DETENTEURS DU DROIT D'AUTEUR ET/OU LES AUTRES PARTIES METTENT LE PROGRAMME A DISPOSITON "EN L'ETAT", SANS AUCUNE GARANTIE DE QUELQUE NATURE QUE CE SOIT, EXPRESSE OU IMPLICITE, Y COMPRIS, MAIS SANS LIMITATION, LES GARANTIES IMPLICITES DE COMMERCIALISATION ET DE L'APTITUDE A UN OBJET PARTICULIER. C'EST LE CONCESSIONNAIRE QUI PREND LA TOTALITE DU RISQUE QUANT A LA QUALITE ET AUX PERFORMANCES DU PROGRAMME. SI LE PROGRAMME SE REVELAIT DEFECTUEUX, C'EST LE CONCESSIONNAIRE QUI PRENDRAIT A SA CHARGE LE COUT DE L'ENSEMBLE DES OPERATIONS NECESSAIRES D'ENTRETIEN, REPARATION OU CORRECTION.
12. EN AUCUN CAS, SAUF SI LA LOI EN VIGUEUR L'EXIGE OU SI UNE CONVENTION ECRITE EXISTE A CE SUJET, AUCUN DETENTEUR DE DROITS D'AUTEUR, OU AUCUNE PARTIE AYANT LE POUVOIR DE MODIFIER ET/OU DE REDISTRIBUER LE PROGRAMME CONFORMEMENT AUX AUTORISATIONS CI-DESSUS, N'EST RESPONSABLE VIS-A-VIS DU CONCESSIONNAIRE POUR CE QUI EST DES DOMMAGES, Y COMPRIS

TOUS DOMMAGES GENERAUX, SPECIAUX, ACCIDENTELS OU INDIRECTS, RESULTANT DE L'UTILISATION OU DU PROGRAMME OU DE L'IMPOSSIBILITE D'UTILISER LE PROGRAMME (Y COMPRIS, MAIS SANS LIMITATION, LA PERTE DE DONNEES, OU LE FAIT QUE DES DONNEES SONT RENDUES IMPRECISES, OU ENCORE LES PERTES EPROUVEES PAR LE CONCESSIONNAIRE OU PAR DES TIERS, OU ENCORE UN MANQUEMENT DU PROGRAMME A FONCTIONNER AVEC TOUS AUTRES PROGRAMMES), MEME SI CE DETENTEUR OU CETTE AUTRE PARTIE A ETE AVISE DE LA POSSIBILITE DE TELS DOMMAGES.

END OF TERMS AND CONDITIONS

H.3 Comment appliquer ces dispositions a vos nouveaux programmes?

Si le concessionnaire dèveloppe un nouveau programme, et s'il en souhaite l'utilisation la plus large possible dans le public, le meilleur moyen d'y arriver est d'en faire un logiciel libre, que tout le monde pourra redistribuer et modifier au titre des prèsentes dispositions. Dans ce but, il convient de rattacher au programme les avis suivants. Le moyen le plus sûr consiste á les rattacher au dèbut de chaque fichier source, pour avertir le plus efficacement possible de l'exclusion de garantie ; et chaque fichier doit comporter au moins la ligne "copyright", et un pointeur indiquant oú est localisèe la totalitè de l'avis.

> *Une ligne pour donner le nom du programme et une idèe de ce qu'il fait.*
> Copyright (C) *yyyy nom de l'auteur*
>
> Ce programme est un logiciel libre ; vous pouvez le redistribuer et/ou le modifier conformèment aux dispositions de la Licence Publique Gènèrale GNU, telle que publièe par la Free Software Foundation ; version 2 de la licence, ou encore (á votre choix) toute version ultèrieure.
>
> Ce programme est distribuè dans l'espoir qu'il sera utile, mais SANS AUCUNE GARANTIE ; sans même la garantie implicite de COMMERCIALISATION ou D'ADAPTATION A UN OBJET PARTICULIER. Pour plus de dètail, voir la Licence Publique Gènèrale GNU .
>
> Vous devez avoir reçu un exemplaire de la Licence Publique Gènèrale GNU en même temps que ce programme ; si ce n'est pas le cas, ècrivez á la Free Software Foundation Inc., 675 Mass Ave, Cambridge, MA 02139, Etats-Unis.

Ajoutez aussi des informations sur le moyen permettant d'entrer en contact avec vous par courrier èlectronique (e-mail) et courrier normal.

Si le programme est interactif, prèvoyez en sortie un court avis, tel que celui qui est prèsentè ci-dessous, lors du dèmarrage en mode interactif.

> Gnomovision version 69, Copyright (C) *yyyy nom de l'auteur*
>
> Gnomovision est livrè absolument SANS AUCUNE GARANTIE ; pour plus de dètail, tapez 'show w'. Il s'agit d'un logiciel libre, et vous avez le droit de le redistribuer dans certaines conditions ; pour plus de dètail, tapez 'show c'.

Les instructions hypothètiques '`show w`' et '`show c`' doivent prèsenter les parties appropriées de la Licence Publique Gènèrale. Bien èvidemment, les instructions que vous utilisez peuvent porter d'autres noms que '`show w`' et '`show c`' ; elles peuvent même correspondre á des clics de souris ou á des èlèments d'un menu, selon ce qui convient á votre programme.

Si nècessaire, vous devrez aussi demander á votre employeur (si vous travaillez en tant que programmeur) ou á votre èventuelle ècole ou universitè, de signer une "renonciation á droit d'auteur" concernant le programme. En voici un èchantillon (il suffit de modifier les noms) :

> Yoyodyne, Inc., par la prèsente, renonce á tout intèrêt de droits d'auteur dans le programme 'Gnomovision' (qui fait des passages au niveau des compilateurs) ècrit par James Hacker.
>
> *signature de Ty Coon*, 1er avril 1989
>
> Ty Coon, President of Vice

La prèsente Licence Publique Gènèrale n'autorise pas le concessionnaire á incorporer son programme dans des programmes propriètaires. Si votre programme est une bibliothéque de sous-programmes, vous pouvez considèrer comme plus intèressant d'autoriser une èdition de liens des applications propriètaires avec la bibliothéque. Si c'est ce que vous souhaitez, vous devrez utiliser non pas la prèsente licence, mais la Licence Publique Gènèrale pour Bibliothéques GNU.

Sauf mention contraire indiquèe plus haut, le prèsent document est soumis aux conditions d'exploitation suivantes :

Ce document peut être reproduit par n'importe quel moyen que ce soit, pourvu qu'aucune modification ne soit effectuèe et que cette notice soit prèservèe.

Annexe I Licence Publique Gènèrale GNU Limitèe

Version 2.1, Fèvrier 1999

This is an unofficial translation of the GNU Lesser General Public License into French. It was not published by the Free Software Foundation, and does not legally state the distribution terms for software that uses the GNU LGPL–only the original English text of the GNU LGPL does that. However, we hope that this translation will help French speakers understand the GNU LGPL better. Voici (http://www.linux-france.org/article/these/lgpl.html) une adaptation non officielle de la Licence Publique Gènèrale Limitèe du projet GNU. Elle n'a pas ètè publièe par la Free Software Foundation et son contenu n'a aucune portèe lègale car seule la version anglaise de ce document dètaille le mode de distribution des logiciels sous GNU LGPL. Nous espèrons cependant qu'elle permettra aux francophones de mieux comprendre la LGPL.

[Ceci est la premiére version publièe de la GPL Limitèe. Elle joue aussi le rôle de successeur de la Licence Publique Gènèrale GNU pour les bibliothéques, version 2, d'oú le numèro de version 2.1.]

I.1 Prèambule

Les licences d'utilisation de la plupart des programmes sont dèfinies pour limiter ou supprimer toute libertè á l'utilisateur. A l'inverse, les Licences Publiques Gènèrales GNU (GNU General Public Licenses) sont destinèes á vous garantir la libertè de partager et de modifier les logiciels libres, et de s'assurer que ces logiciels sont effectivement accessibles á tout utilisateur.

Cette licence, la Licence Publique Gènèrale Limitèe, s'applique á certains programmes de la Free Software Foundation, typiquement les bibliothéques, comme á tout autre programme dont l'auteur l'aura dècidè. Vous pouvez vous aussi l'utiliser, mais nous vous suggèrons de rèflèchir attentivement, en vous fondant sur les explications donnèes ci-dessous, á la meilleure stratègie á employer dans chaque cas particulier, de la prèsente licence ou de la Licence Publique Gènèrale ordinaire.

Libertè des logiciels ne signifie pas nècessairement gratuitè. Nos Licences Publiques Gènèrales sont conçues pour vous assurer la libertè de distribuer des copies des programmes, gratuitement ou non, de recevoir le code source ou de pouvoir l'obtenir, de modifier les programmes ou d'en utiliser des èlèments dans de nouveaux programmes libres, en sachant que vous y êtes autorisè.

Afin de garantir ces droits, nous avons dû introduire des restrictions interdisant á quiconque de vous les refuser ou de vous demander d'y renoncer. Ces restrictions vous imposent en retour certaines obligations si vous distribuez ou modifiez des copies de bibliothéques protègèes par la prèsente Licence. En d'autres termes, il vous incombera en ce cas de :

a. transmettre aux destinataires tous les droits que vous possèdez,

b. expèdier aux destinataires le code source ou bien tenir celui-ci á leur disposition,

c. si vous liez du code á la bibliothéque, leur fournir des fichiers objets complets, de telle sorte qu'ils puissent les lier de nouveau á la bibliothéque aprés l'avoir modifiée et recompilèe,

d. leur remettre cette Licence afin qu'ils prennent connaissance de leurs droits.

Nous protègeons vos droits de deux façons: d'abord par le copyright du logiciel, ensuite par la remise de cette Licence qui vous autorise lègalement á copier, distribuer et/ou modifier la bibliothéque.

Pour protèger chaque auteur, nous stipulons bien que la bibliothéque concernèe ne fait l'objet d'aucune garantie. En outre, si un tiers la modifie puis la redistribue, tous ceux qui en recevront une copie doivent savoir qu'il ne s'agit pas de la version originale afin qu'une copie dèfectueuse n'entache pas la rèputation de l'auteur de la bibliothéque.

Enfin, tout programme libre est sans cesse menacè par des dèpôts de brevets. Nous souhaitons á tout prix èviter que des sociètès puissent dèposer des brevets sur les Logiciels Libres pour leur propre compte, en en restreignant de ce fait les utilisateurs. Par consèquent, nous exigeons que tout dèpôt de brevet accordè á une version de la bibliothéque soit compatible avec la totale libertè d'utilisation exposèe dans la prèsente licence.

La plupart des logiciels du projet GNU, y compris certaines bibliothéques, sont couverts par la Licence Publique Gènèrale ordinaire. La prèsente licence, la Licence Publique Gènèrale GNU Limitèe, concerne un certain nombre de bibliothéques, et différe beaucoup de la Licence Publique Gènèrale ordinaire. Nous couvrons par la prèsente licence certaines bibliothéques afin de permettre á des programmes non libres d'être liès avec ces derniéres.

Quand un programme est liè á une bibliothéque, que ce soit de maniére statique ou par l'utilisation d'une bibliothéque partagèe, l'ensemble forme lègalement parlant un travail combinè, dèrivè de la bibliothéque originale. C'est pourquoi la Licence Publique Gènèrale ordinaire n'autorise une telle èdition de liens que si l'ensemble qui en rèsulte satisfait ses critéres de libertè. La Licence Publique Gènèrale Limitèe est permissive quant aux critéres que doit remplir un code liè avec la bibliothéque en question.

Nous qualifions cette licence de `Limitèe` car les garanties de libertè qu'elle apporte á l'utilisateur sont `limitèes` par rapport á celles de la Licence Publique Gènèrale ordinaire. Elle limite ègalement les avantages que peuvent acquèrir d'autres dèveloppeurs de logiciels libres dans la concurrence avec les programmes non libres. C'est á cause de ces limitations que nous utilisons la Licence Publique Gènèrale ordinaire pour de nombreuses bibliothéques. Cependant, la Licence Limitèe est avantageuse dans certaines circonstances particuliéres.

Par exemple, on observe (rarement) un besoin particulier d'encourager autant que possible l'utilisation d'une certaine bibliothéque, de telle sorte qu'elle devienne un standard de fait. Pour atteindre un tel but, il faut autoriser des programmes non libres á utiliser cette bibliothéque. Un cas plus frèquemment rencontrè est celui oú une bibliothéque libre remplit la même fonction que des bibliothéques non libres et trés rèpandues. Il y a alors peu á gagner á limiter la bibliothéque libre aux logiciels libres, et on utilisera la Licence Publique Gènèrale Limitèe.

Dans d'autres cas, autoriser des programmes non libres á utiliser une bibliothéque particuliére autorise plus de monde á utiliser une grande quantitè de logiciels libres. Par exemple,

la permission d'utiliser la bibliothéque GNU pour le langage C dans des programmes non libres permet á beaucoup plus de gens d'utiliser l'ensemble du systéme d'exploitation GNU, ainsi que sa variante GNU/Linux.

Bien que la Licence Publique Gènèrale Limitèe Limite la libertè des utilisateurs, elle assure que l'utilisateur d'un programme liè avec la Bibliothéque a la libertè et la possibilitè d'exècuter ce programme en utilisant une version modifièe de la Bibliothéque.

Les termes et conditions prècis selon lesquels on peut copier, distribuer, et modifier une telle bibliothéque suivent. Accordez une attention toute particuliére á la diffèrence entre un "travail fondè sur la bibliothéque" et un "travail qui utilise la bibliothéque". Celuilá renferme du code dèrivè de la bibliothéque, alors que celui-ci doit être combinè á la bibliothéque pour être exècutè.

I.2 TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. Le prèsent Accord de Licence s'applique á toute bibliothéque logicielle ou tout autre programme oú figure une note, placèe par le dètenteur des droits ou un tiers autorisè á ce faire, stipulant que ladite bibliothéque ou programme peut être distribuèe selon les termes de la prèsente Licence Publique Gènèrale Limitèe (ègalement appelèe "cette licence"). Chaque personne concernèe par la Licence Publique Gènèrale Limitèe sera dèsignèe par le terme "Vous".

   Une "bibliothéque" signifie une collection de fonctions et/ou de donnèes logicielles prèparèes de maniére á être convenablement lièes avec des programmes d'application (qui utilisent certaines des fonctions et des donnèes) dans le but de former des exècutables.

   Ci-dessous, le terme "Bibliothéque" se rapporte á toute bibliothéque ou oeuvre logicielle distribuèe selon les prèsents termes. Un "travail fondè sur la Bibliothéque" signifie aussi bien la Bibliothéque elle-même que tout travail qui en est dèrivè selon la loi, c'est-á-dire tout ouvrage reproduisant la Bibliothéque ou une partie de cette derniére, á l'identique ou bien modifièe, et/ou traduit dans une autre langue (la traduction est considèrèe comme une modification).

   Le "code source" d'un travail dèsigne la forme de cet ouvrage sous laquelle les modifications sont les plus aisèes. Sont ainsi dèsignès la totalitè du code source de tous les modules composant une bibliothéque logicielle, de même que tout fichier de dèfinition associè, ainsi que les scripts utilisès pour effectuer la compilation et l'installation de la bibliothéque.

   Les activitès autres que copie, distribution et modification ne sont pas couvertes par la prèsente Licence et sortent de son cadre. Rien ne restreint l'utilisation de la Bibliothéque, et les donnèes issues de celle-ci ne sont couvertes que si leur contenu constitue un travail fondè sur la Bibliothéque (indèpendemment du fait d'avoir ètè rèalisè en utilisant la Bibliothéque). Tout dèpend de ce que la Bibliothéque et le programme qui y fait appel sont censès produire.

1. Vous pouvez copier et distribuer des copies conformes et complétes du code source de la Bibliothéque, tel que Vous l'avez reçue, sur n'importe quel support, á condition de placer sur chaque copie un copyright appropriè et une restriction de garantie, de ne

pas modifier ou omettre toutes les stipulations se rèfèrant á la prèsente Licence et á la limitation de garantie, et de fournir avec toute copie de la Bibliothéque un exemplaire de la Licence.

Vous pouvez demander une rètribution financiére pour la rèalisation de la copie et demeurez libre de proposer une garantie assurèe par vos soins, moyennant finances.

2. Vous pouvez modifier votre copie ou vos copies de la Bibliothéque ou partie de celle-ci, ou d'un travail fondè sur cette Bibliothéque, et copier et distribuer ces modifications selon les termes de l'article 1, á condition de Vous conformer ègalement aux conditions suivantes :
   a. Le travail dèrivè doit être lui-même une bibliothéque logicielle.
   b. Ajouter aux fichiers modifiès l'indication trés claire des modifications effectuèes, ainsi que la date de chaque changement.
   c. Distribuer sous les termes de la prèsente Licence l'ensemble de la rèalisation, á tous, et sans frais.
   d. Si une fonctionnalitè de la Bibliothéque modifièe se rèfére á une fonction ou á des donnèes fournies par le programme d'application qui utilise la fonctionnalitè en question sans pour cela utiliser d'argument lors de l'appel de cette derniére, vous devez agir au mieux pour assurer que la fonctionnalitè se comporte correctement et remplit ceux de ses buts qui ont encore un sens lorsqu'une application ne fournit pas la fonction ou les donnèes auxquelles la Bibliothéque se rèfére.

      (Par exemple, si une fonction d'une bibliothéque calcule des racines carrèes, elle a un but absolument bien dèfini, indèpendamment de l'application. Par consèquent, la clause 2d exige que toute fonction ou toutes donnèes fournies par l'application et utilisation cette fonction soient optionnelles : si l'application ne les propose pas, la fonction de calcul de racines carrèes doit encore calculer des racines carrèes.)

   Toutes ces conditions s'appliquent á l'ensemble des modifications. Si des èlèments identifiables de ce travail ne sont pas dèrivès de la Bibliothéque et peuvent être raisonnablement considèrès comme indèpendants, la prèsente Licence ne s'applique pas á ces èlèments lorsque Vous les distribuez seuls. Mais, si Vous distribuez ces mêmes èlèments comme partie d'un ensemble cohèrent dont le reste est fondè sur une Bibliothéque soumise á la Licence, ils lui sont ègalement soumis, et la Licence s'ètend ainsi á l'ensemble du produit, quel qu'en soit l'auteur.

   Cet article n'a pas pour but de s'approprier ou de contester vos droits sur un travail entiérement rèalisè par Vous, mais plutôt d'ouvrir droit á un contrôle de la libre distribution de tout travail dèrivè ou collectif fondè sur la Bibliothéque.

   En outre, toute fusion d'un autre travail, non fondè sur la Bibliothéque, avec la Bibliothéque (ou avec un travail dèrivè de cette derniére), effectuèe sur un support de stockage ou de distribution, ne fait pas tomber cet autre travail sous le contrôle de la Licence.

3. Vous pouvez opter pour la Licence Publique Gènèrale GNU ordinaire pour protèger une copie donnèe de la Bibliothéque. Pour cela, il vous faudra modifier toutes les notes se rèfèrant á la prèsente Licence, pour qu'elles se rèfèrent plutôt á la Licence Publique Gènèrale GNU ordinaire, version 2 (si une version plus rècente de la Licence Publique Gènèrale GNU a vu le jour, vous pouvez alors spècifier cet autre numèro de version si tel est votre dèsir). Ne vous livrez á aucune autre modification dans ces notes.

Une fois que ce changement a ètè effectuè dans une copie donnèe, il est irrèversible pour cette copie, aussi la Licence Publique Gènèrale ordinaire s'appliquera á toutes les copies et tous les travaux dèrivès qui en seront extraits.

Cette option vous servira lorsque vous souhaiterez copier une portion du code de la Bibliothéque dans un programme qui n'est pas lui-même une bibliothéque.

4. Vous pouvez copier et distribuer la Bibliothéque (ou tout travail dèrivè selon les conditions ènoncèes dans l'article 2) sous forme de code objet ou exècutable, selon les termes des articles 1 et 2, á condition de fournir le code source complet de la Bibliothéque, sous une forme lisible par un ordinateur et selon les termes des articles 1 et 2, sur un support habituellement utilisè pour l'èchange de donnèes.

   Si la distribution du code objet consiste á offrir un accés permettant de copier la Bibliothéque depuis un endroit particulier, l'offre d'un accés èquivalent pour se procurer le code source au même endroit satisfait l'obligation de distribution de ce code source, même si l'utilisateur choisit de ne pas profiter de cette offre.

5. On appelle "travail qui utilise la Bibliothéque" tout programme qui n'est dèrivè d'aucune portion de la Bibliothéque, mais qui est conçu dans le but de fonctionner avec cette derniére en l'incluant á la compilation ou á l'èdition de liens. Isolèe, une telle oeuvre n'est pas un travail dèrivè de la Bibliothéque, et sort donc du cadre de cette License.

   Cependant, lier un "travail qui utilise la Bibliothéque" á cette derniére produit un exècutable qui est dèrivè de la Bibliothéque (en ce qu'il en contient des portions), et non plus un "travail qui utilise la Bibliothéque". Cet exècutable est donc couvert par la prèsente License. C'est l'article 6 qui ènonce les conditions de distribution de tels exècutables.

   Dans le cas oú un "travail qui utilise la Bibliothéque" utilise des portions d'un fichier d'en-têtes inclus dans cette derniére, le code objet qui en rèsulte peut fort bien être un travail dèrivè de la Bibliothéque, quand bien même ce n'est pas le cas du code source. Cette prècision prend toute son importance si on peut lier ce travail sans la Bibliothéque, ou si le travail est lui-même une bibliothéque. Le seuil á partir duquel cela prend effet n'est pas exactement dèfini par la loi.

   Si un tel fichier objet n'utilise que des paramétres numèriques, les reprèsentations des structures de donnèes et ce par quoi elles sont lues ou modifièes, ainsi que de petites instructions macros ou fonctions embarquèes (de moins de dix lignes de longueur), alors on pourra utiliser le fichier objet de la maniére que l'on souhaite, qu'il soit ou non, lègalement parlant, un travail dèrivè (mais les exècutables renfermant ce code objet et des portions de la Bibliothéque continuent á être soumis á l'article 6).

   Sinon, si le travail est dèrivè de la Bibliothéque, vous pouvez distribuer le code objet de ce travail selon les conditions de l'article 6. Tout exècutable renfermant ce travail est lui aussi soumis á l'article 6, qu'il soit ou non directement liè avec la Bibliothéque á proprement parler.

6. Le prècèdent article fait exception aux prècèdents. Vous pouvez ègalement combiner ou lier un "travail qui utilise la Bibliothéque" avec cette derniére pour produire un travail contenant des portions de la Bibliothéque, et distribuer ce dernier sous les conditions qui vous sièront, pourvu que ces conditions autorisent la modification de ce travail pour utilisation personnelle, ainsi que l'ingènierie á revers afin de dèboguer ces modifications.

Vous devez fournir avec chaque copie du travail une note trés claire expliquant que la Bibliothéque fut utilisèe dans sa conception, et que la Bibliothéque et son utilisation sont couvertes par la prèsente Licence. Vous devez ègalement fournir une copie de la prèsente Licence. Si le travail, lors de son exècution, affiche des copyrights, vous devez inclure parmi ces derniers le copyright de la Bibliothéque, ainsi qu'une rèfèrence expliquant á l'utilisateur oú il pourra trouver une copie de la prèsente Licence. Vous devez aussi vous conformer á l'un des points suivants :

a. Accompagner le travail avec l'intègralitè du code source pour la Bibliothéque, sous une forme lisible par un ordinateur, ainsi que les èventuelles modifications que vous lui avez apportèes pour rèaliser ce travail (lequel doit être distribuè selon les termes des articles 1 et 2). Si le travail est un exècutable liè avec la Bibliothéque, il vous faut proposer ègalement, sous une forme lisible par un ordinateur, l'ensemble du "travail qui utilise la Bibliothéque", sous forme de code source ou objet, de telle sorte que l'utilisateur puisse modifier la Bibliothéque et effectuer de nouveau l'èdition de liens, afin de produire un exècutable modifiè, renfermant une version modifièe de la Bibliothéque (il est entendu que l'utilisateur qui modifie le contenu des fichiers de dèfinitions de la Bibliothéque ne sera pas forcèment capable de recompiler l'application afin d'utiliser la version modifièe de ces dèfinitions).

b. Utiliser un mècanisme de partage de bibliothéques convenable pour l'èdition de liens avec la Bibliothéque. Un mècanisme convenable est un mècanisme qui : (1) utilise une copie de la bibliothéque dèjá prèsente sur le systéme de l'utilisateur, plutôt que de copier des fonctions de la bibliothéque au sein de l'exècutable, et (2) fonctionnera correctement avec une version modifièe de la bibliothéque, si l'utilisateur en installe une, tant que la version modifièe sera compatible avec la version qui a servi á la rèalisation du travail.

c. Faire une offre ècrite, valable pendant au moins trois ans, proposant de distribuer á cet utilisateur les èlèments spècifiès dans l'article 6a, ci-dessus, pour un tarif n'excèdant pas le coût de la copie.

d. Si le travail est distribuè en proposant un accés á une copie situèe á un endroit dèsignè, proposer de maniére èquivalente, depuis ce même endroit, un accés aux objets spècifiès ci-dessus.

e. Vèrifier que l'utilisateur a dèjá reçu une copie de ces objets, ou que vous la lui avez dèjá envoyèe.

Pour un programme exècutable, la forme requise du "travail qui utilise la Bibliothéque" doit comprendre toute donnèe et tout utilitaire nècessaires pour pouvoir reconstruire l'exècutable. Toutefois, l'environnement standard de dèveloppement du systéme d'exploitation mis en oeuvre (source ou binaire) – compilateurs, bibliothéques, noyau, etc. – constitue une exception, sauf si ces èlèments sont diffusès en même temps que le programme exècutable.

Il est possible que cette clause soit en contradiction avec les restrictions apportèes par les licences d'autres bibliothéques propriètaires qui habituellement n'accompagnent pas le systéme d'exploitation. Une telle contradiction signifie qu'il Vous est impossible d'utiliser ces derniéres en conjonction avec la Bibliothéque au sein d'un exècutable distribuè par Vous.

7. Vous pouvez incorporer au sein d'une même bibliothéque des fonctionnalitès fondèes sur la Bibliothéque, qui forment un travail fondè sur cette derniére, avec des fonctionnalitès issues d'autres bibliothéques, non couvertes par la prèsente Licence, et distribuer la bibliothéque rèsultante, si tant est qu'il est autorisè par ailleurs de distribuer sèparèment le travail fondè sur la Bibliothéque et les autres fonctionnalitès, et pourvu que vous vous acquittiez des deux obligations suivantes:
   a. Accompagner la bibliothéque rèsultante d'une copie du travail fondè sur la Bibliothéque, sans le combiner aux autres fontionnalitès de bibliothéques. Cet ensemble doit être distribuè selon les conditions des articles ci-dessus.
   b. Ajouter á la bibliothéque mixte l'indication trés claire du fait qu'une portion de la bibliothéque est un travail fondè sur la Bibliothéque, et en expliquant oú trouver la version non mèlangèe du même travail.
8. Vous ne pouvez pas copier, modifier, cèder, dèposer ou distribuer la Bibliothéque d'une autre maniére que l'autorise la prèsente Licence. Toute tentative de ce type annule immèdiatement vos droits d'utilisation de la Bibliothéque sous cette Licence. Toutefois, les tiers ayant reçu de Vous des copies de la Bibliothéque ou le droit d'utiliser ces copies continueront á bènèficier de leur droit d'utilisation tant qu'ils respecteront pleinement les conditions de la prèsente Licence.
9. Ne l'ayant pas signèe, Vous n'êtes pas obligè d'accepter la prèsente Licence. Cependant, rien d'autre ne Vous autorise á modifier ou distribuer la Bibliothéque ou quelque travaux dèrivès: la loi l'interdit tant que Vous n'acceptez pas les termes de la prèsente Licence. En consèquence, en modifiant ou en distribuant la Bibliothéque (ou tout travail fondè sur elle), Vous acceptez implicitement tous les termes et conditions de la prèsente Licence.
10. La diffusion d'une Bibliothéque (ou de tout travail dèrivè) suppose l'envoi simultanè d'une licence autorisant la copie, la distribution, l'èdition de liens avec, ou la modification de la Bibliothéque, aux termes et conditions de la Licence. Vous n'avez pas le droit d'imposer de restrictions supplèmentaires aux droits transmis au destinataire. Vous n'êtes pas responsable du respect de la Licence par un tiers.
11. Si, á la suite d'une dècision de Justice, d'une plainte en contrefaçon ou pour toute autre raison (lièe ou non á la contrefaçon), des conditions Vous sont imposèes (que ce soit par ordonnance, accord amiable ou autre) qui se rèvélent incompatibles avec les termes de la prèsente Licence, Vous n'êtes pas pour autant dègagè des obligations liées á celle-ci: si Vous ne pouvez concilier vos obligations lègales ou autres avec les conditions de cette Licence, Vous ne devez pas distribuer la Bibliothéque.

    Si une partie quelconque de cet article est invalidèe ou inapplicable pour quelque raison que ce soit, le reste de l'article continue de s'appliquer et l'intègralitè de l'article s'appliquera en toute autre circonstance.

    Le prèsent article n'a pas pour but de Vous pousser á enfreindre des droits ou des dispositions lègales ni en contester la validitè; son seul objectif est de protèger l'intègritè du systéme de distribution du Logiciel Libre. De nombreuses personnes ont gènèreusement contribuè á la large gamme de logiciels distribuèe de cette façon en toute confiance; il appartient á chaque auteur/donateur de dècider de diffuser ses logiciels selon les critéres de son choix.

12. Si la distribution et/ou l'utilisation de la Bibliothéque est limitèe dans certains pays par des brevets ou des droits sur des interfaces, le dètenteur original des droits qui place la Bibliothéque sous la Licence Publique Gènèrale peut ajouter explicitement une clause de limitation gèographique excluant ces pays. Dans ce cas, cette clause devient une partie intègrante de la Licence.

13. La Free Software Foundation se rèserve le droit de publier pèriodiquement des mises á jour ou de nouvelles versions de la Licence. Rèdigèes dans le même esprit que la prèsente version, elles seront cependant susceptibles d'en modifier certains dètails á mesure que de nouveaux problémes se font jour.

    Chaque version posséde un numèro distinct. Si la Bibliothéque prècise un numèro de version de cette Licence et "toute version ultèrieure", Vous avez le choix de suivre les termes et conditions de cette version ou de toute autre version plus rècente publiée par la Free Software Foundation. Si la Bibliothéque ne spècifie aucun numèro de version, Vous pouvez alors choisir l'une quelconque des versions publiées par la Free Software Foundation.

14. Si Vous dèsirez incorporer des èlèments de la Bibliothéque dans d'autres programmes libres dont les conditions de distribution différent, Vous devez ècrire á l'auteur pour lui en demander la permission. Pour ce qui est des programmes directement dèposès par la Free Software Foundation, ècrivez-nous: une exception est toujours envisageable. Notre dècision sera basèe sur notre volontè de prèserver la libertè de notre Programme ou de ses dèrivès et celle de promouvoir le partage et la rèutilisation du logiciel en gènèral.

NO WARRANTY

15. PARCE QUE L'UTILISATION DE LA BIBLIOTHEQUE EST LIBRE ET GRATUITE, AUCUNE GARANTIE N'EST FOURNIE, COMME LE PERMET LA LOI. SAUF MENTION ECRITE, LES DETENTEURS DU COPYRIGHT ET/OU LES TIERS FOURNISSENT LA BIBLIOTHEQUE EN L'ETAT, SANS AUCUNE SORTE DE GARANTIE EXPLICITE OU IMPLICITE, Y COMPRIS LES GARANTIES DE COMMERCIALISATION OU D'ADAPTATION DANS UN BUT PARTICULIER. VOUS ASSUMEZ TOUS LES RISQUES QUANT A LA QUALITE ET AUX EFFETS DE LA bIBLIOTHEQUE. SI LA BIBLIOTHEQUE EST DEFECTUEUSE, VOUS ASSUMEZ LE COUT DE TOUS LES SERVICES, CORRECTIONS OU REPARATIONS NECESSAIRES.

16. SAUF LORSQU'EXPLICITEMENT PREVU PAR LA lOI OU ACCEPTE PAR ECRIT, NI LE DETENTEUR DES DROITS, NI QUICONQUE AUTORISE A MODIFIER ET/OU REDISTRIBUER LA BIBLIOTHEQUE COMME IL EST PERMIS CI-DESSUS NE POURRA ETRE TENU POUR RESPONSABLE DE TOUT DOMMAGE DIRECT, INDIRECT, SECONDAIRE OU ACCESSOIRE (PERTES FINANCIERES DUES AU MANQUE A GAGNER, A L'INTERRUPTION D'ACTIVITES OU A LA PERTE DE DONNEES, ETC., DECOULANT DE L'UTILISATION DE LA BIBLIOTHEQUE OU DE L'IMPOSSIBILITE D'UTILISER CELLE-CI).

END OF TERMS AND CONDITIONS

I.3 Comment appliquer ces directives á vos nouvelles bibliothéques

Si vous dèveloppez une nouvelle bibliothéque et dèsirez en faire bènèficier tout un chacun, la meilleure mèthode est d'en faire un Logiciel Libre que tout le monde pourra redistribuer et modifier. Vous pouvez atteindre ce but en autorisant la redistribution selon les prèsentes clauses (ou, c'est une autres solution, selon les termes de la Licence Publique Gènèrale ordinaire).

Pour cela, insèrez les indications suivantes dans votre bibliothéque (il est prèfèrable et plus sûr de les faire figurer au dèbut de chaque fichier source; dans tous les cas, chaque module source devra comporter au minimum la ligne de "copyright" et indiquer oú rèsident toutes les autres indications) :

```
une ligne pour donner le nom de la bibliothéque et donner une idèe de sa
finalitè.
Copyright (C) annèe  nom de l'auteur

Cette bibliothéque est libre, vous pouvez la redistribuer et/ou la modifier
selon les termes de la Licence Publique Gènèrale GNU Limitèe publièe par la Free
Software Foundation (version 2 ou bien toute autre version ultèrieure choisie
par vous).

Cette bibliothéque est distribuèe car potentiellement utile, mais SANS AUCUNE
GARANTIE, ni explicite ni implicite, y compris les garanties de
commercialisation ou d'adaptation dans un but spècifique. Reportez-vous á la
Licence Publique Gènèrale GNU Limitèe pour plus de dètails.

Vous devez avoir reçu une copie de la Licence Publique Gènèrale GNU Limitèe en
même temps que cette bibliothéque; si ce n'est pas le cas,ècrivez á la Free
Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307,
Etats-Unis.
```

Ajoutez ègalement votre adresse èlectronique, le cas èchèant, ainsi que votre adresse postale.

Si vous officiez en tant que programmeur, n'omettez pas de demander á votre employeur, votre ètablissement scolaire ou autres de signer une dècharge stipulant leur renoncement aux droits qu'ils pourraient avoir sur la bibliothéque :

```
Yoyodyne, Inc., dèclare par la prèsente ne pas revendiquer de droits sur la
bibliothéque 'Frob' rèalisèe par James Random Hacker.

signature de Ty Coon, 1er Avril 1990
Ty Coon, President of Vice
```

Index des commandes, types et fonctions SQL

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Index conceptuel

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y