Administration et exploitation du SGBDR MySQL

Maurice Libes
libes@com.univ-mrs.fr

Centre d'Océanologie de Marseille
UMS 2196 CNRS

15-18 Octobre 2004

Plan du cours MySQL

- Installation de MySQL
- Lancement du serveur « *mysqld* »
  - Configuration et Paramétrage du serveur
- Création d'une Base de Données
- La gestion des droits sur les Bases
  - La base d'administration de « mysql »

Plan du cours MySQL

- Gestion des comptes utilisateurs
  - Ajout et suppression d'utilisateurs
  - Affectation des permissions pour des utilisateurs
- Design et Création d'une Base
  - Création des tables
  - Les types de données

Plan du cours MySQL

- Manipulation des données dans les tables
  - Insertion, selection, modification, destruction
  - Instructions de transactions et de verrouillage
- Administration, sauvegarde, réparation des Bases
- Instructions d'optimisation des Bases
- Utilisation d'une API PHP ou Perl

Premiers pas

- Installation de MySQL
- Lancement du serveur
- Arrêt du serveur
- Configuration de MySQL
- Connexion au serveur mysqld

Installation

- à partir des RPM
  - rpm -ivh <les paquetages ci dessous>
  - rpm -qa | grep -i MySQL
    - *php-mysql-4.3.2-3mdk*
    - *Libmysql12-4.0.15-1mdk*
    - *perl-Mysql-1.22_19-8mdk*
    - *MySQL-4.0.15-1mdk*
    - *MySQL-client-4.0.15-1mdk*
    - *MySQL-common-4.0.15-1mdk*
- À partir des sources
  - *./configure -help*
  - tar zxvf mysql.tar.gz
  - ./configure –prefix=/opt/mysql –with-charset=latin1
  - make && make install

Installation des binaires

- A partir des binaires proposés sur le site de MySQL
  http://dev.mysql.com/downloads/mysql/4.1.html

```
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 --user=mysql
shell> chown -R root  .
shell> chown -R mysql data
shell> chgrp -R mysql .
shell> bin/mysqld_safe --user=mysql &
```

Installation de la base d'administration

- Dans l'installation à partir des rpm, la base d'administration mysql est installée par défaut dans /var/lib/mysql (paramètre –datadir du daemon mysqld)

- Si on installe à partir des binaires, ou que l'on souhaite recréer une nouvelle bases d'administration
  - Lancer le script
    - ./bin/mysql_install_db

Lancement

- Créer un utilisateur et un groupe « *mysql* » qui servira à définir l'identité du serveur “mysqld”. Il faut faire appartenir les bases à cet utilisateur afin qu'il ait toutes les permissions dans les tables

  - ***$ chown -R mysql.mysql /opt/mysql/var/***

- Lancer le daemon serveur mysqld
  - *$ /opt/mysql/bin/mysqld_safe --basedir=/opt/mysql --datadir=/opt/mysql/var/ --user=mysql --log --log_isam --log-update&*

    Starting mysqld daemon with databases from /opt/mysql/var/

Lancement de MySQL

$ ***/etc/rc.d/init.d/mysql -?***

Utilisation : /etc/rc.d/init.d/mysql {***start*** | ***stop*** | ***status*** | ***reload*** | ***restart***}

$ ps ax |grep mysql

1944 ? S 0:00 /bin/sh ***/usr/bin/mysqld_safe*** **--datadir**=/var/lib/mysql –pid-file=/var/lib/mysql/localhost.pid –**log --log-update**

1977 ? S 0:00 ***/usr/sbin/mysqld*** --***basedir***=/ --***datadir***=/var/lib/mysql --***user***=mysql --*pid-file*=/var/lib/mysql/localhost.pid –skip-locking

1978 ? S 0:00 /usr/sbin/mysqld --basedir=/ --datadir=/var/lib/mysql --user=mysql --pid-file=/var/lib/mysql/localhost.pid –skip-locking

--log-update permet de loguer toutes les changements dans une base et de réaliser des backup incrémentaux. Toute l'histoire d'une base peut etre logguée dans */var/lib/mysql/localhost.log*

Lancement de MySQL

Comment est lancé le serveur dans /etc/rc.d/init.d/mysql?

$ cat /etc/rc.d/init.d/mysql

- gprintf "Starting MySQL Server"
- $bindir/***mysqld_safe*** *--datadir=$datadir* --pid-file=$pid_file 2>&1 | logger -t mysqld_safe & success

$ opt/mysql/bin/mysqld_safe --basedir=/opt/mysql/ --datadir=/opt/mysql/var/ --user=mysql --log --log_isam --log-update

Starting mysqld daemon with databases from /opt/mysql/var/

Pour connaître l'ensemble des options de mysqld

- man mysqld
- man mysqld_safe

Arrêt du serveur MySQL

- *$ mysqladmin -u root -p shutdown*
- *$ /etc/rc.d/init.d/mysql stop*

- *$ killall mysqld*
- *$ mysqld_pid=`cat /opt/mysql/var/localhost.pid`*
- *$ kill $mysqld_pid*

Configuration de MySQL

- Options du serveur mysqld :
  - Soit au lancement sur la ligne de commande
  - Soit dans un fichier de conf « /etc/*my.cnf* »
  - *mysqld lit ses options dans les sections [mysqld] et [server].*
  - *mysqld_safe lit ses options dans les sections [mysqld], [server], [mysqld_safe], et [safe_mysqld]*
  - *mysql.server lit ses options dans la partie [mysqld] et [mysql.server]*

- Pour voir les options que l'on peut placer dans ces sections : ***mysqld –help***

Options disponibles de my.cnf

- [client]
  - host=localhost
  - user=user_mysql
  - password=toto
- [mysqld]
  - datadir=/var/lib/mysql
  - socket=/var/lib/mysql/mysql.sock
  - skip-innodb
- [mysql.server]
  - user=mysql
  - basedir=/var/lib
- [safe_mysqld]
  - err-log=/var/log/mysqld.log
  - pid-file=/var/run/mysqld/mysqld.pid

Connexion au serveur

- Connexion au serveur MySQL
  - mysql -u root -h localhost
  - *La première fois pas de mot de passe pour se connecter !!*
- Mettre *tout de suite* un mot de passe pour l'utilisateur « root » de MySQL !

  /opt/mysql/bin/mysqladmin -u root password 'titi'
- *Quelques commandes intéressantes:*
  - *$ mysqladmin variables | mysql> show variables*
  - *$ mysqladmin version*
  - *$ mysqladmin status | mysql> show status*

Utiliser une base courante

- Pour spécifier l'utilisation d'une base particulière :
  - 1. Au lancement du client
    - Mysql -u root -p CHOCOLATS
  - 2.1 Sous l'interpreteur mysql
    - Mysql> Use CHOCOLATS
    - Mysql> select * from CLIENTS
  - 2.2 En notation « pointée » en préfixant les tables par le nom de base [BASE.table]
  - Mysql> select * from CHOCOLATS.CLIENTS;

Gestion des comptes utilisateurs

- Gestion des Accès à la Base

Sécurité des accès

- MySQL utilise sa propre base d'administration "*/var/lib/mysql/mysql*" pour gérer la sécurité des accès aux autres bases.
- Seul le user " root " de MySQL doit avoir accès à cette base!
- La base « mysql » utilise 5 tables pour décider « qui » a la permission de faire « quoi » sur quelle « base », à partir « de quelle machine »

- ***La base « mysql » contient 5 tables d'administration***
  - ***$ ls /var/lib/mysql/mysql/***
  - ***user.frm , host.frm , db.frm***
  - ***columns_priv.frm func.frm tables_priv.frm***

Sécurité des accès

- La table « user » contient les informations de sécurité qui s'appliquent à tout le serveur.
- La table « host » contient les machines qui ont le droit d'accéder au serveur
- Les tables db, tables_priv et columns_priv contiennent les droits d'accès pour une base, une table ou une colonne.

- L'ordre des contrôles d'accès est le suivant (cf doc. Texte word)

Les permissions MySQL

- *mysql> SHOW PRIVILEGES;*

- SELECT : droit d'effectuer des recherches avec “select »
- INSERT : droit d'effectuer des insertions avec « insert »
- *UPDATE* : droit d'effectuer des mises à jour avec « update »
- *DELETE* : droit d'effectuer des destructions d'enregistrements dans des tables
- *INDEX* : droit de créer ou détruire des index de tables
- *ALTER* : droit de modifier la structure des tables avec « alter »
- *CREATE* : droit de créer des bases ou des tables avec « create »
- USAGE : droit de se connecter au serveur, sans rien faire d'autre
  - *(utile uniquement pour changer le mot de passe de connexion)*
- *LOCK :* droit de verrouiller/déverrouiller des tables

Les permissions MySQL

- *Drop* : droit de détruire des bases ou des tables
- *Grant* : permet d'affecter des droits et permission à un utilisateur
- *References* : (droit lié aux « foreign keys », inutilisé encore dans les version actuelle de mysql..)

- Privilèges globaux, ne s'appliquent pas à UNE base particulière
  - *Reload* : permission de relancer le serveur mysql et d'ecrire les tables sur disques.
  - *Shutdown* : droit d'arrêter le serveur “mysqld »
  - *Process* : droit de contrôler les processurs utilisateurs.
  - *File* : droit d'écrire ou lire dans des fichiers ascii avec les commandes « *load data* » et « *into outfile* »

Allouer des permissions:GRANT

- Les permissions des tables se manipulent avec les commandes sql GRANT et REVOKE ou bien avec l'utilitaire *mysqlaccess*
- **GRANT** permet de créer un utilisateur, lui allouer des droits et changer son mot de passe

  *GRANT priv_type [(column_list)] [, priv_type [(column_list)]] ON {tbl_name | * | *.* | db_name.*} TO user [IDENTIFIED BY [PASSWORD] 'password']*

- mysql> *GRANT all ON test.* TO momo IDENTIFIED BY "titi "*;
- mysql> *GRANT select,insert ON mysql.* TO momo@localhost identified by "titi" [with GRANT OPTION];*

les privilèges « file », « reload », "shutdown" sont globaux pour tout le serveur et non pas pour une base particulière

  - mysql> *grant file on *.* to momo@localhost ;*

affichage des droits d'accès

Les exemples précédents insèrent le user « momo » avec son mot de passe, et la machine autorisée, dans la table « user », de la base mysql, sans aucun droit. Les droits sont placés pour la base mysql dans la table « db »

- Vérification des permissions pour un utilisateur

Les 2 privilèges SELECT,INSERT sont placés dans la table « DB » pour une base particulière

- *Mysql*> ***select * from db where user= »momo »***

| localhost | mysql | momo | **Y** **| Y** | N | N | N | N | N | N
| N | N | N | N |

- *Mysql*> ***select * from user where user= »momo »***

| localhost | momo | 398e7500242ab90c | N | N | N | N | N | N | N
| N | N | N | N | N | N | N | N | N | N
| N | N | N | N | |

- mysql> **SHOW GRANTS** FOR root;
- mysql> **SHOW GRANTS** FOR 'root'@'localhost' ;
- $ **mysqlaccess -t -d CHOCOLATS -u momo -U root -P**

2 commandes utiles

- Changer le mot de passe d'un utilisateur
  - GRANT USAGE ON *.* TO momo@localhost IDENTIFIED BY 'titi'
- Créer un super-utilisateur (~root)
  - GRANT ALL PRIVILEGES ON *.* TO supermomo@localhost IDENTIFIED BY 'titi' WITH GRANT OPTION;
  - L'option GRANT donne le droit d'utiliser la commande GRANT et donc de permettre à un utilisateur d'en créer d'autre
  - (NB: ne pas multiplier les « super-utilisateurs !!)

Oter des permissions: REVOKE

- REVOKE permet de supprimer des droits pour des utilisateurs

REVOKE priv_type [(column_list)] [, priv_type [(column_list)]]
*ON {tbl_name | * | *.* | db_name.*}*
FROM user [, user]

*mysql> REVOKE ALL on *.* FROM momo ;*
mysql> REVOKE ALTER,DELETE,DROP on mysql. FROM momo*
mysql> REVOKE GRANT OPTION on mysql. FROM momo@localhost*
mysql> REVOKE SELECT on mysql. FROM 'momo'@'pcml.dom.fr'*

Enlever un utilisateur

- à partir de la version 4.1.1 (nécessité d'enlever tous les privileges avant de détruire l'utilisateur)

```
mysql> SHOW GRANTS FOR toto@localhost
mysql> REVOKE ALL PRIVILEGES FROM
  toto@localhost
mysql> DROP USER toto;
```

- Pour des versions antérieures

```
mysql> DELETE FROM mysql.user WHERE
  user='user_name' and Host='host_name';
```

- *Les permissions ont été placées dans la table mysql.*, il faut les recharger en mémoire pour le serveur*

```
mysql> FLUSH PRIVILEGES;
```

Création & destruction d'une base

- Création d'une base = créer un répertoire
  1. $ mysqladmin -u root -p create *NOMDEBASE*
  2. $ mkdir /var/lib/*NOMDEBASE && chown mysql /var/lib/NOMDEBASE*
  3. $ mysql -u root -p
     - mysql> CREATE DATABASE *ma_base*
- *Destruction d'une base*
  - *$ mysqladmin -p drop NOMDEBASE*
    - *mysql> DROP DATABASE ma_base*

TP : création d'une Base CONFISEUR

- Regarder le script *script_creation_tables_base_confiseurs.txt*
- Se créer une base vierge
  - */opt/mysql/bin/mysqladmin -u root -p create CHOCO*
- *Créer les tables en batch avec le script*
  - */opt/mysql/bin/mysql -u root -p < script_creation_tables_base_confiseurs.txt*
- Connexion en tant que “root”… Quelques vérifications
  - *Mysql> show tables*
  - *Mysql> desc BOITES*
  - *Mysql> select * from BOITES*

TP : Mettre des droits sur la Base

- *Créer plusieurs utilisateurs et affecter des droits différents à ces utilisateurs par GRANT et REVOKE*

  - *Un utilisateur qui aura tous les droits sur la Base*
  - *Un utilisateur qui n'aura que le droit de sélectionner*
  - *Un utilisateur qui aura le droit d'écrire dans les tables et les modifier et récupérer les sélections dans un fichier*

  - *Conclusions?*
  - *GRANT all ON test.* TO momo**@localhost** IDENTIFIED BY "titi ";*
  - *Ou*
  - ***insert into host (host,Db,Select_priv) values ("localhost","CHOCOLATS","Y");***
  - *delete from user where host="localhost" and user="";*

TP : Les droits sur la Base

- Modifier un mot de passe pour un utilisateur
- Enlever le droit update et file pour l'utilisateur qui les a.

Maintenance et Administration de MySQL

- Structures des tables
- Instructions de maintenance des Tables
  - BACKUP TABLE
  - RESTORE TABLE
  - REPAIR TABLE
  - CHECK TABLE
  - ANALYZE TABLE
  - OPTIMIZE TABLE

La structure des Tables MySQL

- Une table au format MyISAM (séquentiel indexé) se décompose en 3 fichiers :
  - table.MYD
  - table.MYI
  - table.frm

- Le fichier .MYD contient les données réelles (D-ata)
- Le fichier .MYI contient les informations sur les clés et index des tables. Toutes les informations pour retrouver rapidement les données sont dans le fichier .MYI. Ce fichier est capital pour optimiser les accès aux tables.
- Le fichier .frm contient la structure de la table elle même (n'a pas d'impact sur les performances)

Structure des tables

- Le fichier .MYI contient les index, et est le plus important pour les performances des accès aux tables.
- La commande « *myisamchk* » est entièrement consacrée aux opérations sur ce fichier
- Les opérations vont concerner:
  - L'analyse et le listage des clés,
  - La réparation de tables endommagées, la re-création et destruction des clés...

Backup des bases

- Les bases sont des répertoires, et les tables des fichiers... les opérations de sauvegarde et de backup sont donc simples!

Avec un serveur en activité, pour assurer des copies sécurisées:

- *mysql>LOCK TABLES;*
- *mysql>FLUSH TABLES*

(assure que tous les index et données sont bien écrits avant de commencer la copie.)

- Hors du contrôle du serveur, et serveur arrété, on peut faire régulièrement (cron) une simple copie des bases:

$ rsync -av /var/lib/mysql/unebase /opt/SAUVEBASE/

Backup des bases

*L'utilitaire **mysqldump** permet de faire des sauvegardes complètes en ascii*

- Backup : *$ mysqldump --opt database [table] > backup-file.sql*
  - *$ /opt/mysql/bin/mysqldump -u root -ptoto CHOCOLAT BOITES*
- Restoration:
  - *$ mysql database < backup-file.sql*

Backup avec mysqlhotcopy

- Sauvegarde d'une base complète avec toutes les tables dans –datadir (/var/lib/mysql/)

$ /usr/bin/mysqlhotcopy --debug -u root –password='titi' CHOCOLATS CHOCOLATS-SAV

- Sauvegarde de la table CLIENTS de la base CHOCOLATS dans un répertoire extérieur à / var/lib/mysql

$ /usr/bin/mysqlhotcopy --debug --addtodest -u root --password='aaa' CHOCOLATS./CLIENTS/ "/home/momo"

BACKUP / RESTORE

- *Mysql> BACKUP TABLE COMMANDES to "/var/lib/mysql/SAUVE/";*
  - *(NB1:le user « mysql » doit avoir le droit d'écrire dans le répertoire indiqué)*
  - *NB2:cette commande n'est plus supportée… Il est préférable d'utiliser la commande mysqlhotcopy*
  - *NB3: Ne sauvegarde que .frm et .MYD, les index doivent etre reconstruits (avec **REPAIR TABLE matable USE_FRM**)*
  - *NB4: Ne fonctionne que sur des tables myisam*
- *La commande RESTORE TABLE sert à remettre en service des tables sauvegardées par BACKUP TABLES*

  mysql> RESTORE TABLE BOITES FROM "/var/lib/mysq/SAUVE/";

Maintenance de Tables

- Pour vérifier et réparer des tables MyISAM
  - CHECK TABLE
  - REPAIR TABLE
- Pour optimiser des tables MyISAM :
  - OPTIMIZE TABLE
  - ANALYZE TABLE
- L'utilitaire « myisamchk » hors du contrôle du serveur

Les problèmes avec les Tables MyISAM

- Le format des tables MyISAM est très sûr. Mais on peut avoir des tables corrompues dans quelques cas malchanceux:
  - le serveur mysqld s'arrête pendant une écriture
  - Arrêt brutal de l'ordinateur
  - Erreur hardware (contrôleur de disque…)
  - Utilisation d'un programme externe (myisamchk) sur une table qui est en train d'être modifiée par le serveur en même temps..
- Les symptômes lors de l'accès à une table
  - *Incorrect key file for table: '...'. Try to repair it*
- Utiliser
  - ***CHECK TABLE***
  - ***REPAIR TABLE***
  - ***Myisamchk***

Verification des tables : check

- En cas de problème tel que les données semblent corrompues, ou si l' on obtient constamment des erreurs lors d'accès à une table, il faut essayer de réparer la table suspecte avec l'utilitaire de ligne de commande myisamchk ou les commandes SQL CHECK TABLE et REPAIR TABLE.

- Vérification de la structure des tables... calcul et recherche des index, tests de corruption sur les tables MyISAM

  CHECK TABLE BOITES {QUICK | FAST | MEDIUM | EXTENDED | CHANGED}

- identique à :
  - *$ myisamchk --medium-check /path/tbl_name*
  - *myisamchk --medium-check /opt/mysql/var/CHOCOLAT/BOITES*

Verification des tables : check

- En cas de corruption de tables, les problèmes viennent le plus souvent du fichier des index et non pas des données.
- Pour vérifier simplement une table il faut utiliser l'option QUICK. Si MYSQL trouve un erreur dans la table des données, la table est marquée comme étant « Corrupted » et ne peut plus être utilisée jusqu'à ce qu'elle soit réparée!!!

- FAST and CHANGED doivent être utilisée dans des scripts pour vérifier régulièrement l'etat des tables
- MEDIUM passe en revue chaque ligne pour vérifier que les liens vers les index sont corrects. Calcule un checksum pour chaque ligne et le compare avec un checksum calculé sur les clés
- EXTENDED doit être utilisé après détection d'une erreur qui pourrrait subsister

Réparation des tables : repair

REPAIR TABLE répare une table potentiellement corrompue. En cas de désastre, REPAIR TABLE devrait réussir à récupérer les données à partir de la table. Dans la majeure partie des cas on ne devrait pas avoir besoin de cette commande.

REPAIR [LOCAL | NO_WRITE_TO_BINLOG] TABLE tbl_name... *[QUICK] [EXTENDED]* [USE_FRM]

Cette commande a le même effet que :

- *$ myisamchk [--recover | safe_recover] tbl_name*
- *Cette commande ne fonctionne QUE sur des tables MyISAM*

Réparation des tables : repair

- *Si l'option QUICK est donnée, REPAIR TABLE essaye de réparer uniquement l'arbre des index. La commande est equivalente à*
  - *myisamchk --recover --quick.*

Si l'option EXTENDED est donnée, MySQL recalcule et re-crée les index ligne par ligne. Équivalent à :

- *myisamchk –safe-recover.*

- *A partir de MySQL 4.0.2, l'option USE_FRM de REPAIR TABLE permet de recréer le fichier des index `.MYI' à partir du fichier `.frm'*

Optimisation des tables

- ANALYZE [LOCAL | NO_WRITE_TO_BINLOG] TABLE tbl_name

- Cette instruction analyse et stocke la distribution des clés d'une table. (Pendant l'analyse la table est « verrouillée » (lock table))
- L'instruction fonctionne sur les tables MyISAM et InnoDB

Pour les tables MyISAM l'instruction est équivalente à:

  - *myisamchk -a --description –verbose /path/vers/latable*

- MySQL utilise la distribution des clés stockées pour décider de l'ordre dans lequel seront faites les jointures des tables.

Optimisation des tables

- ANALYSE TABLE recalcule les clés et utilise la distribution des clés stockées pour décider de l'ordre dans lequel seront faites les jointures des tables.
- Il est recommendé d'exécuter « *myisamchk -a* » régulièrement lors de la création d'une base (insertion des données dans les tables). En effet la majeure partie des données sont insérées au début de la création des bases.
- En exécutant une analyse de table « *myisamchk -a* » régulièrement lors de l'insertion de données, on permet à ce que les données soient conservées le plus efficacement possible

Optimisation des tables

OPTIMIZE TABLE réorganise une table en récupérant les espace vides et en défragmentant la table.

Syntaxe: OPTIMIZE [LOCAL | NO_WRITE_TO_BINLOG] TABLE tbl_name [, tbl_name] ...

OPTIMIZE TABLE doit être utilisé dans les cas où :

- on a détruit une grande quantité de données dans les tables.
- On a fait des changements de types sur les données (lignes de longueur variables comme VARCHAR, TEXT ou BLOB)

Les enregistrements détruits sont conservés dans une liste chainée et les futures insertions de données pourront réutiliser les anciennes positions d'enregistrements détruits..

Optimisation des tables

OPTIMIZE TABLE effectue les opérations suivantes :

1. répare la table en récupérant la place laissée par des lignes vides
2. trie les index, s'ils ne sont pas triés
3. met à jour les statistiques des tables

L'utilitaire myisamchk

- Commande entièrement consacrée à l'analyse et la maintenance (réparation) du fichier d'index .MYI et de données .MYD
- Myisamchk peut être utilisé pour obtenir de l'information sur les tables, pour les vérifier, les réparer ou les optimiser.
- On préconise d'effectuer des backups des bases avant d'effectuer des réparations sur les fichiers d'index.
- La plupart des actions de myisamchk sont également réalisées par les commandes SQL précédentes (*check, repair, analyse optimize*) sous le contrôle du serveur mysqld.

L'utilitaire myisamchk

- Myisamchk est une commande qui est exécutée sous le shell et n'est donc pas sous le contrôle du serveur MySQL. Les modifications engendrées nécessiteront alors un « reload » du serveur ou un FLUSH des tables .
- L'avantage des commandes SQL est que les opérations effectuées sont sous le contrôle du serveur (les locks de tables sont effectués, vidage des tampons)
- Avec myisamchk, on doit s'assurer que le serveur n'utilise pas les tables en même temps.
- Si le serveur tourne, il faut forcer l'écriture des tables qui sont bufferisées en mémoire en utilisant FLUSH TABLES. Il faut s'assurer que personne n'utilise les tables pendant qu'on exécute myisamchk. CHECK TABLE est préférable à myisamchk pour vérifier les tables.

Myisamchk pour «vérifier»

Sans option : simple vérification des erreurs dans les tables

$ myisamchk /opt/mysql/var/CHOCOLAT/*.MYI

- *--check, -c* : cherche des erreurs dans la table (option par défaut)
- *--check-only-changed, -C* : vérifie les tables qui ont changées depuis la dernière vérification
- *--extend-check, -e* : Vérification complète et approfondie des tables.Opération d'autant plus longue que la table a beaucoup d'index. Option à utiliser dans des cas extrèmes. myisamchk --medium-check devrait suffire pour trouver des erreurs dans la table.
- *--extend-check* : nécessite beaucoup de mémoire. On peut mettre la variable key_buffer_size a une plus grande valeur pour accélérer les calculs.

Myisamchk pour «vérifier»

- *--force, -f* provoque une réparation automatique des tables si myisamchk trouve des erreurs. Le type de réparation est celui spécifié dans l'option. --repair ou -r.
- *--information, -i* : affiche des informations statistiques sur la table
- *--medium-check, -m* : vérification plus rapide qu'avec l'option --extend-check .
- *--update-state, -U* : stocke les informtions dans le fichier .MYI pour indiquer quand la table a été vérifiée. Option inéressante pour tirer bénéfice de l'option --check-only-changed. On ne doit pas utiliser cette option si le serveur mysqld utilise la table sans l'option --skip-external-locking .

Myisamchk pour chercher des erreurs

- **myisamchk tbl_name :** trouve 99.99% des erreurs.

- **myisamchk -m tbl_name**
  trouve 99.999% des erreurs. Vérification des index de chaque ligne : Calcul d'un checksum pour chaque clé des lignes et vérifie que cette valeur de checksum est égale au checksum des clés dans l'arbre des index.

- **myisamchk -e tbl_name :**
  Vérification complète des données (``extended check''). Vérification en lecture de toutes les clés de chaque ligne pour vérifier qu'elles pointnt sur la bonne ligne.

- **myisamchk -e -i tbl_name**
  idem ci dessus mais avec l'option “-i” affiche des informations statistiques supplémentaires.

Myisamchk pour «réparer»

--recover , -r : répare une table endommagée. Défragmente les tables. Cette option convient à presque tous les cas de problèmes. Essayer l'option « -o » si les problèmes persistent. Si la machine a beaucoup de mémoire, on peut augmenter la valeur du paramètre *« sort_buffer_size »*

- /opt/mysql-4.1/bin/mysqladmin -u root -p variables |grep sort_buffer

--safe-recover, -o

réparation avec une autre méthode, qui lit toutes les lignes dans l'ordre et mets à jour tous les index basés sur les lignes trouvées. Méthode plus lente qu'avec -r et qui utilise plus de place disque

- Essayer d'abord les réparations avec -r puis avec -o si échec.

Myisamchk pour «analyser»

--analyse , -a : Analyse la distribution des clés. Cela améliore les performances de jointure en permettant à l'optimiseur de mieux choisir l'ordre dans lequel sera faite la jointure des tables et quelle clé sera utilisée.

- *SHOW KEYS FROM tbl_name* .

--description, -d : donne les principales informations sur la table. Les informations les plus importantes sont :

```
Data records:          18  Deleted blocks:          0
Recordlength:                     320
Key Start Len Index   Type
1   2     4   unique  char
```

--sort-index, -S

tri l'arbre des index. Cela optimize les recherches par clé et les rends plus rapides.

Myisamchk en cas de crash

- Etre sur que personne n'utilise les tables à travers le serveur, pendant qu'on utilise myisamchk. Faire un backup avant d'utiliser myisamchk.
- faire ***mysqladmin flush-tables*** avant de commencer à vérifier les tables. Sinon arréter le serveur mysqld pendant les vérifications et réparations de tables.
- La plupart des problemes surviennent sur les fichiers d'index .MYI ou de données .MYD.
- Myisamchk crée une copie du fichier`.MYD' ligne à ligne. Il termine la réparation en détruisant l'ancien fichier .MYD et en renommant le nouveau
- Avec l'option *--quick*, myisamchk ne crée pas le fichier .MYD temporaire , il considère que le fichier .MYD est correct, et ne recrée que le fichier des index
- On peut spécifier l'option --quick 2 fois. Dans ce cas , myisamchk essaye de resoudre les erreurs en modifiant le fichier `.MYD' .

Conclusions sur myisamchk

- Commande utile pour conserver des Bases de données sans anomalies et réparer les erreurs éventuelles :

- Inspecter : Exécuter quotidiennement myisamchk -d -c -i -s conserver le compte rendu et le consulter
- Réparer : Lors de problème de crash (disque, serveur, machine..) analyser les tables des bases et passer myisamchk -a -r -e
- Défragmenter : Lancer régulièrement myisamchk -r -o … d'autant plus souvent que les mises à jour des tables (insert, update, delete) sont intenses et fréquentes
- Optimiser,analyser : myisamchk-a -d

TP sur maintenance et analyse des tables

- Faire une sauvegarde avec *mysqldump*
- Faire une sauvegarde avec *mysqlhotcopy*
- Faire une sauvegarde avec *backup*

- Utiliser les commandes
  - Check
  - Repair
  - Analyse
  - Optimize
  - mysqlchk

Surveillance du serveur

- Le serveur MySQL gère les connexion des clients. Pour chaque client mysqld crée un processus fils (un *thread*) qui prend en charge les requêtes de ce client. Il y a donc autant de *threads* que de connexions.
- Chaque *thread* accède aux données des tables en fonction des requêtes de son client
- Le serveur gère un ensemble de base de données (chacune étant un répertoire placé sous la racine du serveur, indiquée par l'option –datadir au lancement du serveur)
- Mysql fournit un ensemble d'utilitaires pour surveiller l'état du serveur :mysqladmin, kill, show

Mysqladmin – KILL

- Charge et status du serveur
  - $ mysqladmin -u user -p [*status | extended-status | variables]*

Uptime: 36707 Threads: 2 Questions: 65 Slow queries: 0 Opens: 26 Flush tables: 3
Open tables: 9 Queries per second avg: 0.002

- Surveillance des threads
  - $ mysqladmin -u user -p *processlist*
  - ***Mysql> SHOW PROCESSLIST***

Id	***User***	***Host***	***db***	*Command*	*Time*	*State*	*Info*
21	*momo*	*localhost*	*CHOCOLAT*	*Sleep*	*2*		
22	*root*	*localhost*		*Query*	*0*		*show processlist*

- Tuer un thread
  - $ mysqladmin -u user -p *kill 21*
  - *mysql> KILL 21*

Statistiques du serveur

$ mysqladmin -p status

Enter password:

Uptime: 27266 Threads: 1 Questions: 22 Slow queries: 0 Opens: 12 Flush tables: 1 Open tables: 6 Queries per second avg: 0.001

- *Statistiques TRES détaillées sur l'état du serveur, nombre de fichiers ouverts, de tables ouvertes, taux d'E/S en octets, nombre de clés utilisées, nombre de query ...*
  - SHOW STATUS
- Toutes les variables qu'utilise le serveur
  - SHOW VARIABLES
    - Identique à : *$ mysqladmin variables*

SHOW

- SHOW DATABASES : montre les bases existantes
  - *Equivalent à* ***/opt/mysql/bin/mysqlshow -u root -ptoto***
- SHOW TABLES *[like "expr"]* : montre les tables d'une Base
  - *Equivalent à* ***/opt/mysql/bin/mysqlshow CHOCOLAT -u root -ptoto***
- SHOW CREATE TABLE BONBONS ;
- SHOW COLUMNS : affiche les colonnes d'une table données
  - **SHOW [FULL] COLUMNS FROM *tbl_name [LIKE 'pattern']***
- ***DESCRIBE*** [nom_de_table]
  - *Identique à SHOW FIELDS FROM BOITES;*
- /opt/mysql/bin/mysqlshow CHOCOLAT CLIENTS -u root -ptoto
- SHOW PROCESSLIST : affiche les threads actifs du serveur

SHOW

- **SHOW PRIVILEGES** : affiche la liste des privilèges système de mysql (à partir de la version 4.1)
- **SHOW INDEX** : affiche les informations sur les index des tables
  - **SHOW KEYS** : identique
  - $ **mysqlshow -k** nom_base nom_table : identique
- **SHOW GRANTS** : affiche les privilèges d'un utilisateur
  - **SHOW GRANTS FOR momo@localhost;**

| GRANT USAGE ON *.* TO 'momo'@'localhost' IDENTIFIED BY PASSWORD '398e7500242ab90c' |

| GRANT ALL PRIVILEGES ON `CHOCOLAT`.* TO 'momo'@'localhost' WITH GRANT OPTION |

  - *équivalent à :* ***mysqlaccess -t -d CHOCOLAT -u momo -U root -P***

SHOW

- Statistiques TRES détaillées sur l'état du serveur, nombre de fichiers ouverts, de tables ouvertes, taux d'E/S en octets, nombre de clés utilisées, nombre de query …
  - SHOW STATUS

- Toutes les variables qu'utilise le serveur
  - SHOW VARIABLES
    - Identique à : *$ mysqladmin variables*

Prise en compte des modifications d'accès : FLUSH

- Au lancement du serveur les tables sont chargées en mémoire, pour des accès plus rapide. Si on modifie des tables, sur disque l'image en mémoire ne correspond pas. L'opération “flush” indique au serveur de mettre en correspondance les données sur le disque et celles qui sont en mémoire.
- Il est donc nécessaire de recharger les tables en mémoire après modification des tables de la base mysql

  - *$ mysqladmin reload = $ mysqladmin flush-privileges*
  - *$ mysqladmin flush-tables* (ecrit les tables sur disques)
  - *$mysqladmin flush-logs* (ecrit les fichiers de log)
  - *$mysqladmin refresh* (les 2 précédentes)

  ou

  - *mysql> FLUSH PRIVILEGES;*

FLUSH

- FLUSH est la commande à utiliser pour vider les caches mémoires de diverses structures, sur disque. Il faut avoir le privilège « *reload* » pour utiliser la commande.

FLUSH HOSTS : vide la mémoire cache de la tables « HOSTS ». A utiliser dans le cas de certaines erreurs (*Host ... is blocked*), ou quand on dépasse le nombre d'erreurs de connexion défini par *max_connect_errors*

FLUSH LOGS : écriture des fichiers de log. ferme et réouvre tous les fichiers de « log ». Le numéro de fichier en extension est incrémenté de 1

FLUSH

- FLUSH PRIVILEGES : recharge en mémoire (et donc prend en compte) les privilèges de la table des permissions de la base mysql *(inutile avec GRANT)*,

- FLUSH TABLES *[nom_table]* : écriture des mises à jours différées.. la commande ferme toutes les tables en cours d' utilisation et les réouvre . La commande détruit également la mémoire cache des résultats issus des requêtes SQL comme la commande RESET QUERY CACHE.
- FLUSH STATUS : remise à zéro de la plupart des variables de mysql. Utile en cas de debuggage de certaines requetes SQL.
- Ces commandes SQL, FLUSH, réalisent les mêmes fonctions que la commande *mysqladmin* avec les options *flush-hosts, flush-logs, flush-privileges, flush-status, ou flush-tables*.

RESET

- RESET est utilisé pour annuler l'état de plusieurs opération du serveur. RESET agit comme un FLUSH généralisé en vidant la mémoire cache de toutes les structures utilisées par le serveur, sur le disque. Il faut avoir le privilège RELOAD pour utiliser RESET
  - RESET MASTER : détruit les fichiers de log binaires et en crée un nouveau
  - RESET QUERY CACHE enlève les résultats de commande SQL du « query cache »
  - RESET SLAVE : indique au serveur « esclave » d'oublier sa position de replication dans les logs binaires

Les « logs » du serveur

- Avoir lancé le serveur avec certaines options:
  - *–log-update (jusqu'à la version 4 de MySQL, les logs sont en ascii)*
  - *--bin-log (à partir de la version 5 les logs sont binaires et peuvent être lus avec l'utilitaire mysqlbinlog)*

- permet de loguer toutes les modifications faites sur les bases *(alter, insert, update)*

Les différents « logs » du serveur

Différentes options de "logs" au lancement du serveur :

- **--log-error** : trace des problèmes rencontrés au lancement, exécution du serveur mysqld dans un fichier .err.
- **--log-isam** : trace de tous les changements effectués sur les tables ISAM.
- **--log** :Trace normale des connexions des clients et des exécutions de requêtes SQL.
- **--log-update** : Trace des instructions qui provoquent des modifications des données (ALTER, UPDATE..). Cette trace sera inopérante dans les prochaines versions >=5.0
- **--log-bin** : Trace des instructions qui provoquent des modifications des données, en format binaire (à partir de la version 4.1.3). Cette trace est utilisée également pour la réplication des bases.
- **--log-slow-queries** : trace des requetes qui prennent plus de "long_query_time" seconds pour s'exécuter, oui n'utilisent pas d'index.

Les « logs » du serveur

Par défaut, tous les fichiers de logs sont créés dans le répertoire donné par –datadir « /var/lib/mysql ».

- Si on donne le nom du fichier au lancement --log-update [=file_name] option, mysqld créée un fichier qui contient toutes les instructions SQL qui ont modifié les données
- Sans nom de fichier le nom du fichier par défaut est le nom de la machine */opt/mysql/var/localhost.log*
  - On peut forcer le serveur à fermer et ré-ouvrir les fichiers de logs par les instructions :
    - mysql> FLUSH LOGS ;
    - mysqladmin flush-logs
    - mysqladmin refresh.

TP sur Surveillance du serveur

DEUXIEME PARTIE

- Les types de données mysql
- Les commandes MySQL
  - Création de tables
  - Création d'index
  - Manipulation de données
    - Select, insert, replace, update, delete
- Les instructions de transaction

Les types de données de MySQL

- types numériques
- types Dates
- type caractères et TEXT
- type BLOB (binary Long Object)
- type ENUM
- type SET

Les types numériques

TINYINT	1 octet	non ANSI
SMALLINT	2 octets	ANSI
MEDIUMINT	3 octets	non ANSI
INTEGER(M) ZEROFILL	4 octets	ANSI
BIGINT	8 octets	ANSI
FLOAT(M,D) ZEROFILL	4 octets	ANSI
DOUBLE {PRECISION}	8 octets	ANSI
REAL	8 octets	ANSI
NUMERIC (M,D)	M,(D+2 si M<D)	ANSI
DECIMAL (M,D)	M,(D+2 si M<D)	ANSI

- M,D signifie : numérique de taille maximal M avec D décimales
  - 3,14156 <=> 6,5
- ZEROFILL = entier non signé

Les types de DATE

DATE

Une date représente un intervalle entre le '1000-01-01' et le '9999-12-31'. MySQL affiche les DATE au format 'YYYY-MM-DD' Toutefois des fonctions de conversion comme « date_format() permettent de changer de format.

DATETIME

permet de combiner une date et un horaire. L'affichage se fait au format AAAA-MM-JJ HH:MM:SS entre le *1000-01-01 00:00:00' et '9999-12-31 23:59:59*.

TIMESTAMP[(M)]

Ce type stocke une date et un horaire sous forme d'un nombre de secondes écoulées depuis le 1er janvier 1970. La syntaxe est timestamp (M) où M est la longueur de l'affichage (14 par defaut)

Les types de DATE

TIME

A time représente un horaire seulement. Le format est en HH:MM:SS .

YEAR[(2|4)]

Format pour avoir une date sur 2 ou 4 positions? Sur 4 position on représente des dates dans la gamme de 1901 à 2155. Sur 2 positions les valeurs permises vont de 70 à 69 (1970 à 2069) L'affichage se fait dans le format YYYY .

Les types « caractères »

- Le type CHAR(n) [BINARY] : permet de stocker des chaines de caractères d'une taille maximale fixe de n caractères (avec n<255).
  - MySQL complète la chaine avec des blancs si la taille réelle est inférieur à n
  - MySQL tronque la chaine réelle si la longueur réelle > n
  - L'option *BINARY* permet de prendre en compte la casse des caractères (majuscules/minuscules). Avec BINARY, «AAA » sera différent de « aaa »

- Le type VARCHAR(n) [BINARY] : permet de stocker des chaînes de longueur variable

Les types « caractères » longs

- Les type *BLOB (binary large object) et TEXT* permettent de représenter des chaines de longueur variables .
  - TINYBLOB , TINYTEXT : *longueur max 2^8 car.*
  - BLOB , TEXT : *longueur max 2^16 car.*
  - MEDIUMBLOB, MEDIUMTEXT : *longueur max 2^24 car.*
  - LONGBLOB , LONGTEXT : *longueur max 2^32 car.*

- Les comparaisons sur des valeurs de type TEXT ne distinguent pas majuscules et minuscules.
- TEXT et BLOB sont des VARCHAR de grandes tailles, avec l'option BINARY activée pour le type BLOB

CHAR ou VARCHAR ?

- Avec CHAR les enregistrements ont une taille fixe (=somme des tailles de chaque attribut). Enregistrement facile à gérer puisqu'on peut le référencer par son numéro d'ordre. La position d'un enr de taille fixe = numéro d'ordre * taille
- Accès direct facile, Réparations faciles avec myisamchk

- Avec VARCHAR, BLOB ou TEXT les enr ont une taille variable.
- Lors de l'insertion d'un N-uplet on doit calculer la taille exacte des enr d'après le nombre d'octets de chaque attribut. La taille totale est stockée au début de l'enregistrement
- Gestion plus complexe par chainage des enregistrements
- Fragmentation inévitable, et performances moindre en temps d'accès, mais gain de place

Les types ENUM et SET

- ENUM : permet d'indiquer un type énuméré dont les instances prennent leur unique valeur parmi un ensemble explicitement spécifié
  - Exemple: *ENUM ('bleu', 'blanc', 'rouge')*

  MySQL contrôle contrôle qu'une valeur rentrée appartient bien à l'ensemble énuméré.

- SET : identique à ENUM, mais un attribut de type SET peut prendre plusieurs valeurs parmi l'ensemble énuméré.

Les COMMANDES MySQL

- La création des bases
- Destruction de bases
- Création des tables
- Destruction des tables
- Modifications de la structure des tables
- Création d'index

Création/Destruction de Base

- mysql> CREATE DATABASE *nomBase*
  - *Crée un répertoire vide dans /var/lib/mysql*
- mysql>DROP DATABASE [IF EXISTS] *nomBase*
  - *Détruit tous les fichiers (toutes les tables) d'un base*
  - *Le script s'interrompt si la base n'existe pas, sauf si l'option IF EXISTS est spécifiée*

Création de Tables

- *CREATE [TEMPORARY] TABLE [IF NOT EXISTS] nom_table*
  [(create_definition,...)]
  [table_options] [select_statement]

```
CREATE TABLE CLIENTS (
  codeclient` int(11) AUTO_INCREMENT,
  nom char(20) NOT NULL default,  prenom char(20) NOT NULL default ',
  rue char(30) default NULL,  ville char(15) default NULL,
  region char(20) default NULL,  codepostal char(5) default NULL,
  pays char(15) default NULL,  telephone char(24) default NULL,
  PRIMARY KEY  (nom,prenom),
  INDEX (profession)
) TYPE=MyISAM
```

Destruction de Tables

- ***mysql> DROP TABLE nom_table1, table2 [IF EXISTS] ;***
  - Il est nécessaire d'avoir le privilege DROP sur toutes les tables.
  - Prudence avec cette commande car toutes les tables et les données sont définitivement perdues.

Modification de la structure des Tables

La création d'un schéma de table n'est qu'une étape dans la vie de la base. On peut être amené à rajouter, enlever des attributs, des index, modifier le type des attributs…

- ***mysql> ALTER TABLE nom_table1 ACTION ;***
  - avec ACTION =
    - ADD COLUMN nom_col clause_de_creation
    - ADD INDEX nomindex (col1, col2)
    - ALTER nom_col SET DEFAULT valeur
    - CHANGE
    - MODIFY nom_col clause_de_creation
      - alter table COMMANDES modify codecommande int(8);
    - DROP INDEX nom_index
    - DROP COLUMN nom_col
    - RENAME new_table

Exemples de modification de structure de tables

Rajout d'un attribut dans une table :

- ***mysql> ALTER TABLE DETAILCOMMANDES ADD datecommande DATETIME;***

- ***Mysql> ALTER TABLE DETAILCOMMANDES MODIFY datecommande DATETIME NOT NULL;***

- **Plus fréquement on peut être amené à rajouter ou modifier des index dans les tables**

Les Index

- Dans une table volumineuse la recherche *séquentielle* d'un enregistrement est une opération pénalisante en terme de temps de recherche
- Les index (contenus dans .MYI) permettent d'améliorer les temps de recherche en accédant directement aux enregistrements du fichier de données (.MYD). Les index sont des valeurs (adresses) auxquelles on a rangé les enregistrements dans le fichier de données. Grâce aux index on accède alors aux enregistrements directement
- On a intérêt à créer des index pour les attributs sur lesquels seront effectués de nombreuses recherches

les index à la Création de Tables

- *PRIMARY KEY* : la clé primaire est synonyme d'index à valeur unique non nulle qui sera utilisé plus particulièrement dans les jointures de tables. Il y a une seule PK par table. (la PK a un statut particulier (unique et not null), mais c'est un index comme les autres)
- *KEY* : KEY est un synonyme de INDEX.
- *INDEX nom_index (nom,prenom)* : création d'un index sur un attribut donnée, ou une concaténation d'attributs
- *UNIQUE* : un index à valeur unique. Une insertion de 2 valeurs identiques sur un attribut déclaré « UNIQUE » échouera.

Création / Destruction d'index a posteriori

- Si on a oublié des index lors de la création des tables de la base, on peut rajouter des index pendant la vie la base de données, sans avoir à reconstruire les tables

  - *mysql> CREATE INDEX nom_index ON nom_table (colonne1, colonne2)*
  - *mysql> CREATE INDEX codeclient ON COMMANDES (codeclient);*

- Destruction des index a posteriori
  - *Mysql> DROP INDEX nom_index ON nom_table*
  - *Mysql> ALTER TABLE nomtable DROP INDEX nom_index*

Quelques optimisations en temps de recherche

- Récupérer les emplacement vides (défragmenter) avec myisamchk -r ou optimize table
- Choix de type de données compact (SMALLINT au lieu de INTEGER, chaines de tailles fixes au lieu de chaines variables)
- Éviter les valeur nulles, déclarer les attributs NOT NULL le plus souvent possible
- Préférer CHAR à VARCHAR (réfléchir vitesse? Ou place disque?)
- Utiliser, quand cela est possible des tables temporaires avec l'option TEMPORARY ou des tables de type HEAP
  - Créer des index adéquats sur les attributs qui le “nécessitent»

TP sur Creation, Destruction Modification

- Créer une base « disque »
- Création des 3 tables de type InnoDB
  - Artiste
  - Disque
  - chanson
- Modifications de la structure des tables
- Création d'index

Les COMMANDES MySQL

- La manipulation des données des tables:
  - Selection
  - Insertion
  - Modification
  - Destruction

SELECT

- SELECT est utilisé pour obtenir des enregistrements venant d'une ou plusieurs tables. Le support des commandes UNION et des sous-requêtes est disponibles depuis MySQL 4.0 et 4.1

- SELECT [STRAIGHT_JOIN] [DISTINCT] *listeattributs*
  - INTO OUTFILE 'nomfichier'
  - FROM *nomdetable*
  - WHERE *condition*
  - GROUP BY *nomattribut*
  - HAVING *condition*
  - ORDER BY *nomattribut* [ASC | DESC]
  - LIMIT *debut,nbligne*
- *STRAIGHT_JOIN indique que la jointure des tables doit se faire dans l'ordre indiqué*

Jointure avec SELECT

- Les jointures permettent d'exprimer des requêtes portant sur des données réparties dans plusieurs tables
- On donne la liste des tables dans la clause FROM, et on exprime les critéres de rapprochement dans la clause WHERE
- La plupart des jointures s'expriment par une égalité entre la clé primaire d'une table et la clé étrangère correspondante dans l'autre table.
- L'optimiseur de MySQL fait au mieux pour utiliser les meilleurs index pour minimiser le nombre de lignes parcourues.

- Pour forcer l'optimiseur à utiliser un ordre spécifique de jointure dans une commande SELECT,il faut ajouter l'attribut STRAIGHT_JOIN à la clause SELECT .

Jointure avec SELECT

- ***Quel est le nom et le prix des boites commandés par quantité de 2 ?***

mysql> SELECT D.codecommande,D.codeboite,D.quantite,B.nom_boite, B.prix_boite from DETAILCOMMANDES D, BOITES B where D.quantite=2 and D.codeboite=B.code_boite group by D.codeboite;

- ***Détail des commandes de Mr Fadiman (3 jointures de tables) ?***

select C.nom,C.codeclient,M.codecommande, from CLIENTS C,COMMANDES M where C.nom='Fadiman' and C.codeclient=M.codeclient ;

select C.nom,C.codeclient,M.codecommande,D.codeboite,D.quantite from CLIENTS C,COMMANDES M,DETAILCOMMANDES D where C.nom='Fadiman' and C.codeclient=M.codeclient and M.codecommande=D.codecommande;

select C.nom,C.codeclient,M.codecommande,D.codeboite,D.quantite,B.nom_boite,B.prix_boite from CLIENTS C,COMMANDES M,DETAILCOMMANDES D,BOITES B where C.nom='Fadiman' and C.codeclient=M.codeclient and M.codecommande=D.codecommande and D.codeboite=B.code_boite;

EXPLAIN

- EXPLAIN, permet d'identifier les index qui seront utilisés lors d'une requête SELECT et de mettre en évidence les index qu'il faut ajouter pour accélérer les commandes SELECT.

- EXPLAIN indique egalement l'ordre dans lequel seront faites les jointures. Cela permet également de vérifier que l'optimiseur fait les jointures dans un ordre vraiment optimal.

- Pour forcer l'optimiseur à utiliser un ordre spécifique de jointure dans une commande SELECT,il faut ajouter l'attribut STRAIGHT_JOIN à la clause SELECT .
- L'optimiseur peut se tromper et ne pas choisir les index corrects. L'utilitaire *myisamchk –analyse* permet d'étudier la distribution des index par colonnes et de choisir le meilleur index

EXPLAIN

```
explain select
    C.nom,C.codeclient,M.codecommande,D.codeboite,D.quantite,B.nom_boite,B.p
    rix_boite from CLIENTS C,COMMANDES M,DETAILCOMMANDES D,BOITES
    B where C.nom='Fadiman' and C.codeclient=M.codeclient and
    D.codecommande=M.codecommande and D.codeboite=B.code_boite;
| table | type   | possible_keys | key     | key_len | ref           | rows | Extra       |
+-------+--------+---------------+---------+---------+---------------+------+-------------+
| C     | ref    | PRIMARY       | PRIMARY |      20 | const         |    1 | Using where |
| D     | ALL    | PRIMARY       | NULL    |    NULL | NULL          | 1359 |             |
| M     | eq_ref | PRIMARY       | PRIMARY |       4 | D.codecommande |    1 | Using
    where |
| B     | eq_ref | PRIMARY       | PRIMARY |       4 | D.codeboite    |    1 |             |
+-------+--------+---------------+---------+---------+---------------+------+-------------+
4 rows in set (0.00 sec)
```

EXPLAIN

la colonne type indique l'efficacité de la jointure

- **System**

  La table a une seule ligne (c'est une table système). C'est un cas spécial du type de jointure “const” (constante).

- ***const***

  La table a au plus une ligne correspondante, qui sera lue dès le début de la requête. Les tables “const” sont très rapides, car elles ne sont lues qu'une fois. “const” est utilisé lorsque on compare toutes les parties d'une clé PRIMAIRE avec une constante « ***WHERE email='libes@com.univ-mrs.fr'*** »

- ***eq_ref***

  C'est le meilleur type de jointure possible, à l'exception des précédents. Il est utilisé lorsque toutes les parties d'un index sont utilisées par la jointure, et que l'index est UNIQUE ou PRIMARY KEY. ***eq_ref*** est utilisé pour les colonnes indexées, qui sont comparées avec l'opérateur =. L'élément comparé doit être une constante ou une expression qui utilise les colonnes de la table qui est avant cette table.

EXPLAIN

- ***ref***
  - ***ref*** est utilisé si la jointure n'utilise que le préfixe de gauche de la clé, ou si la clé n'est pas UNIQUE ou PRIMARY KEY (en d'autres termes, si la jointure ***ne peut pas*** sélectionner qu'une seule ligne en fonction de la clé). Si la clé qui est utilisée n'identifie que quelques lignes à chaque fois, la jointure est bonne. ref peut être utilisé pour les colonnes indexées, qui sont comparées avec l'opérateur =.
- ***range***
  - Seules les lignes qui sont dans un intervalle donné seront lues, en utilisant l'index pour sélectionner les lignes. La colonne key indique quel est l'index utilisé. key_len contient la taille de la partie de la clé qui est utilisée. La colonne ref contiendra la valeur NULL pour ce type. range peut être utilisé lorsqu'une colonne indexée est comparée avec une constante comme =, <>, >, >=, <, <=, IS NULL, <=>, BETWEEN ou IN.

EXPLAIN

- ***Index***

  Identique à ALL, hormis le fait que seul l'arbre d'index est étudié. C'est généralement plus rapide que ALL, car le fichier d'index est plus petit que le fichier de données. Cette méthode peut être utilisée lorsque la requête utilise une colonne qui fait partie d'un index.

- ***ALL***

  Une analyse ***complète*** de la table est faîte pour chaque combinaison de lignes issue des premières tables. Ce n'est pas bon si la première table n'est pas une jointure de type const et c'est très mauvais dans les autres cas. Normalement vous pouvez éviter ces situations de ALL en ajoutant des index basée sur des parties de colonnes.

EXPLAIN

possible_keys

La colonne possible_keys indique quels index MySQL va utiliser pour trouver les lignes. Cette colonne est totalement dépendante de l'ordre des tables. Cela signifie que certaines clés de la colonne possible_keys pourraient ne pas être utilisées dans d'autres cas d'ordre de tables. Si cette colonne est vide, il n'y a pas d'index pertinent. On peut améliorer les performances en examinant la clause WHERE pour voir si des colonnes sont susceptibles d'être indexée.

key

La colonne key indique l'index que MySQL va décider d'utiliser. Si la clé vaut NULL, aucun index n'a été choisi. Pour forcer MySQL à utiliser un index listé dans la colonne possible_keys, utilisez USE KEY/IGNORE KEY dans votre requête. See section 14.1.7 Syntaxe de SELECT.

key_len

La colonne key_len indique la taille de la clé que MySQL a décidé d'utiliser. Cela vous indique combien de partie d'une clé multiple MySQL va réellement utiliser.

ref

La colonne ref indique quelle colonne ou quelles constantes sont utilisées avec la clé key, pour sélectionner les lignes de la table.

rows

La colonne rows indique le nombre de lignes que MySQL estime devoir examiner pour exécuter la requête.

EXPLAIN

- Dans la table D aucun index n'est utilisé... parcours séquentiel de 1359 lignes
- Rajout d'un index sur le codeclient dans la table COMMANDES

mysql> ALTER table COMMANDES ADD INDEX client (codeclient);

Query OK, 407 rows affected (0.01 sec)

Records: 407 Duplicates: 0 Warnings: 0

```
| C     | ref     | PRIMARY         | PRIMARY |       20 | const                       |    1 | Using where |
| M     | ref     | PRIMARY,client | client  |        5 | C.codeclient                |    1 | Using where |
| B     | ALL     | PRIMARY         | NULL    |     NULL | NULL                        |   18 |             |
| D     | eq_ref | PRIMARY,boite  | PRIMARY |        8 | M.codecommande,B.code_boite |    1
    |             |
+-------+--------+----------------+---------+---------+-----------------------------+------+-------------+
```

JOIN – jointure externe

- Les jointures externes permettent de trouver les données présentes dans la table de gauche, qui N'ont PAS de valeurs correspondantes dans la table de droite
- Si pour une valeur de la table de gauche on trouve une valeur pour la table de droite, la jointure s'effectue normalement, sinon les attributs de la table de droite sont affichés à NULL

- ***Quels sont les clients qui n'ont PAS fait de commandes?***

  *select CL.codeclient,CL.nom,C.codecommande from CLIENTS CL **LEFT JOIN** COMMANDES C **ON** C.codeclient=CL.codeclient ;*

  *select CL.codeclient,CL.nom,C.codecommande from CLIENTS CL **NATURAL LEFT JOIN** COMMANDES C WHERE C.codecommande **IS NULL**;*

S'écrit aussi

*select CL.codeclient,CL.nom,C.codecommande from CLIENTS CL **NATURAL left join** COMMANDES C;*

TP sur SELECT et EXPLAIN et jointures

- 2 grosses jointures à écrire et à améliorer:
  - Indiquer les Noms prenoms des clients qui ont commandés des bonbons M13 (W06) ?
  - Trouver les commandes qui ont été passées pour Mr Smith?
- 2 jointures externes
  - Qui n'a pas fait de commandes?
  - trouver les bonbons qui NE SONT PAS dans des boites

UNION

- Dans un SGBDR les tables sont considérées comme des ensembles sur lesquel on peut effectuer les opération d'UNION, INTERSECTION..etc..
- UNION permet de combiner les résultats de plusieurs instructions SELECT en un seul ensemble résultant . *UNION est supporté à partir de MySQL 4.0.0*

- *Mysql> SELECT codecommande FROM COMMANDES UNION SELECT codecommande from DETAILCOMMANDES;*

Requêtes Imbriquées

- Les requetes SELECT / WHERE permettent de retrouver des valeurs scalaires.Avec le mot clé « IN » on peut retrouver une valeur parmi un ensemble
  - SELECT * from BOITES WHERE code_boite IN ('ROMA', 'SWEE');
  - SELECT * from BOITES WHERE code_boite NOT IN ('ROMA', 'SWEE');

- Les requêtes imbriquées sont une généralisation de cette construction. Au lieu de donner un ensemble de valeurs constantes (« en dur »), on le construit dynamiquement avec une 2ème requête imbriquée dans la première.
  - Les requêtes imbriquées sont supportées à partir de MySQL 4.1

REQUETES IMBRIQUEES

- A partir de la version 4.1 on pourra écrire

```
SELECT * FROM BOITES WHERE code_boite IN
    (SELECT  code_boite FROM DETAILCOMMANDES WHERE
      quantite="1");
```

- Avant la version 4.1, ce type de requête imbriquée peut toutefois s'écrire encore avec des jointures

```
SELECT *  FROM BOITES B, DETAILCOMMANDES D where
   B.code_boite=D.codeboite AND D.quantite="3";
```

TP requête imbriquées

- Trouver les boites qui ont été commandées par quantité supérieure à 2

```
select * from BOITES where code_boite in (select codeboite from
    DETAILCOMMANDES where quantite>"2");
```

- trouvez les boites dans lesquelles il ya des bonbons à base de noix??
  - Ecrire la requete en requete imbriquée
  - La même SANS requete imbriquée

INSERT

- Pour insérer une ou plusieurs lignes dans une table

- ***Insertion de données explicites dans les colonnes désignées***
  - *Mysql> INSERT INTO BOITES (code_boite,nom_boite,taille) VALUES ("MOMO","orange au chocolat", 110);*
  - *Mysql> INSERT INTO BOITES (code_boite,nom_boite,taille) VALUES ("MOMO","orange au chocolat", 110), ("GOOD","praline", 245);*

- ***Insertion de valeurs retournées par une instruction « SELECT », copie de lignes d'une table dans une autre***
  - Mysql> *INSERT INTO BOITES (code_boite,nom_boite,taille) SELECT code_boite,nom_boite,taille FROM CHOCOLATS2.BOITES where CHOCOLATS2.BOITES.taille > "400";*

Insertion de données à partir d'un fichier

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE
   'file_name.txt'
   [REPLACE | IGNORE]     INTO TABLE tbl_name
   [FIELDS                    Pour indiquer les séparateurs de colonnes
      [TERMINATED BY '\t']
      [[OPTIONALLY] ENCLOSED BY '']
      [ESCAPED BY '\\' ]
   ]
   [LINES       [STARTING BY '']     [TERMINATED BY '\n']    ]
   [IGNORE number LINES]    Pour sauter « n » lignes en début
   [(col_name,...)]                    Pour sauter la colonne col_name
```

LOAD DATA infile

- Permet d'insérer des données dans une table à partir d'un fichier de texte dont les colonnes sont séparées par un certain séparateur
- Le fichier *boites.txt* doit avoir une ligne pour chaque enregistrement inséré dans la table
- Le mot clé LOCAL indique que le fichier se situe sur la même machine que le client. Sinon mysql cherche le fichier sur le serveur, dans le répertoire *–datadir*
- Par défaut les colonnes sont séparées par des tabulations, les lignes terminées par \n
  - ***LOAD DATA LOCAL INFILE "/home/momo/boites.txt" INTO TABLE BOITES;***
  - ***LOAD DATA INFILE "/home/momo/bonbons.txt" REPLACE INTO TABLE BONBONS;***

REPLACE

- REPLACE est identique à INSERT mais porte sur des lignes qui existent déjà (si la ligne existe déjà INSERT échoue)
- Avec REPLACE si un enregistrement a la même valeur que l'enregistrement que l'on insère, l'ancien enregistrement est détruit et le nouveau est inséré.
- Si l'enregistrement n'existe pas (valeur de Clé primaire absente), il est inséré comme avec INSERT), les colonnes que l'on ne mentionne pas sont initialisées à NULL !!

- *Mysql> REPLACE INTO BOITES (code_boite,nom_boite,taille,description_boite) VALUES ("SWEE","Doux et Amer","500"," tres bonne boite de bonbons");*
- Une méthode rapide pour recharger toute une table à partir d'un fichier
  - *LOAD DATA INFILE "/home/momo/cours-mysql/bonbons.txt" REPLACE INTO TABLE BONBONS;*

UPDATE

- Pour mettre à jour des attributs existants dans une table,

- Toute ligne qui satisfait à l'expression de recherche de la clause « WHERE » est modifiée.
  - Mysql> UPDATE BOITES SET taille="400", nom_boite="tendres cremes" WHERE code_boite="SWE2";
- On peut utiliser le nom d'une colonne comme valeur sur elle même
  - UPDATE BOITES SET taille=taille+2 WHERE code_boite="SWE2";

- NB:Il faut le privilege « UPDATE » pour exécuter cette instruction

DELETE

- Pour détruire un enregistrement
  - Mysql> DELETE FROM BOITES WHERE code_boite="SWEU";

- ATTENTION: Si on ne met pas de clause de sélection « WHERE » toute la table entière est écrasée, et une nouvelle table vide est recréée.

- Lors de la destuction d'enregistrement, on ne détruit qu'un pointeur dans une liste chainée. L'espace des lignes détruites n'est PAS physiquement restitué au système. L'espace peut etre réoccupé par des instructions d'insertion futures.
- Pour récupérer physiquement la place disque il faut lancer
  - OPTIMIZE TABLE ou Myisamchk -r

TP insert, delete, load data

Les tables de type InnoDB

- Les apports des tables de type InnoDB
  - Les contraintes d'intégrités grâce aux clés étrangères (Foreign Keys)
  - Les requêtes transactionnelles

Les tables de type InnoDB

- L'organisation des tables est différente des tables MyISAM. On ne trouve plus que les fichier .frm. Les fichiers d'index .MYI et de données .MYD n'existent plus
- InnoDB dispose d'un buffer spécial pour mettre en cache les données et les index en mémoire centrale. InnoDB stocke les tables et index dans un “tablespace”
- Dans */etc/my.cnf* section *[mysqld]*
  - spécifier innodb_data_file_path pour donner les noms et tailles de fichiers de données. *innodb_data_file_path = /ibdata/ibdata1:10M:autoextend*
  - spécifier innodb_data_home_dir pour donner le répertoire de données, Par défaut les fichiers sont créés dans le dossier de données de MySQL.
- Le comportement par défaut est de créer un fichier de données auto-croissant de 10Mo appelé ibdata1
- Pour ne pas utiliser les tables InnoDB, on peut rajouter l'option *skip-innodb* dans le fichier d'options MySQL *my.cnf*.

Les tables de type InnoDB

- InnoDB fournit à MySQL un gestionnaire de table transactionnelle avec validation (*commits*), annulations (*rollback*) et possibilité de restauration après crash. InnoDB utilise un verrouillage de lignes, et fournit des lectures cohérentes sans verrous. Ces fonctionnalités accroissent les possibilités d'utilisation simultanées des tables, et les performances.
- Les tables InnoDB sont les premières tables MySQL qui supportent les contraintes de clés étrangères (FOREIGN KEY)
- InnoDB a été conçu pour maximiser les performances lors du traitement de grandes quantités de données. Son efficacité processeur n'est égalée par aucun autre moteur de base de données.
- Le support des tables InnoDB est inclus par défaut à partir de MySQL 4.0 (ou en compilant le code source à partir des versions MySQL-3.23

Les tables de type InnoDB

On ne peut pas détruire un fichier dans le “tablespace”! Pour diminuer la taille du fichier tablespace (ibdata1), il faut faire de la manière suivante :

1. mysqldump to dumper toutes les tables InnoDB
   /opt/mysql/bin/mysqldump -u root -ptoto CHOCOINNO BOITES
2. Arréter le serveur *(mysql shutdown)*
3. Détruire tous les fichiers ”tablespace”
4. Configurer un nouveau tablespace.

innodb_data_file_path = /ibdata/ibdata1:988M;/disk2/ibdata2:50M:autoextend

5. Redémarrer le serveur.
6. Importer les fichiers de données précédemment dumpés

Creation de tables de type InnoDB

- Il suffit de spécifier type=InnoDB lors de la création de table

```
Mysql> create table IF NOT EXISTS BOITES (
                code_boite char(4) primary key,
               INDEX (nom_boite char(40)),   taille int(5),
               description_boite varchar(255),  prix_boite decimal(8,2),
                qte_en_stock int(5),    rupture_stock tinyint
               ) type = InnoDB;
```

La table et l'index sont créés dans le tablespace InnoDB qui consiste en les fichiers de données qu'on a spécifié dans `my.cnf'.

MySQL crée un fichier `BOITES.frm' dans le répertoire CHOCOINNO (dans --datadir). En interne InnoDB rajoute à son dictionnaire une entrée pour la table CHOCOINNO/BOITES. On peut créer des tables de noms identiques dans d'autres bases InoDB.

myql> show table status from CHOCOINNO like 'BOITES';

Les clés étrangères « FOREIGN KEYS »

- Une « clé étrangère » est l'identifiant unique (la clé primaire) d'une table de l'autre coté d'une relation 1,N
- Les foreign keys (FK) permettent d'indiquer quels sont les attributs qui font référence à une ligne dans une autre table
- Les FK sont implémentées dans les tables de type InnoDB (la syntaxe est acceptée dans les tables myisam, mais n'est pas utilisée)
- Les clés étrangères SQL sont utilisées pour assurer la cohérence des données dans les tables liées par des relations, et non pas pour faire jointures
- Les clés étrangères permettent d'éviter d'insérer des valeurs incohérentes dans la base (valeur orpheline = 1 valeur de clé dans une relation qui n'existerait pas dans la table correspondante).

Les clés étrangères « FOREIGN KEYS »

- La seule chose que MySQL ne fait pas (avec les types autres que InnoDB), est de s'assurer que la clef qu'on utilise dans une table existe bien dans la ou les tables que vous référencez
- En l'absence de vérification du coté du serveur, l'application doit se charger des vérifications. Elle doit s'assurer que les lignes ne sont pas orphelines. Il faut aussi pouvoir rattraper une erreur au milieu d'une opération multiple.
- MySQL permet aux développeurs de BD le choix de leur approche. Si on n'a pas besoin des clés étrangères, et que on veut éviter leur surcoût, il est préférable d'utiliser des tables MyISAM. Les tables MyISAM sont extrêmement rapides pour les applications qui font essentiellement des opérations INSERT et SELECT.

Les clés étrangères « FOREIGN KEYS »

- Avec les foreign keys dans les tables de type INNODB, MySQL permet une vérification centralisée de contraintes, ce qui rend inutile l'exécution de ces vérifications du coté de l'application qui insère les données. Cela élimine la possibilité que d'autres applications ne fassent pas les vérifications de la même façon que les autres.
- Ces avantages entrainent un coût supérieur pour le serveur de bases, qui de ce fait doit effectuer les tests. **Avec les clés étrangères, c'est le serveur MySQL qui vérifie les contraintes d'intégrités (i.e les valeurs pour une FK d'une relation doivent exister comme PK dans l'autre relation**
- Les vérifications supplémentaires par mysql affectent donc les performances. Certaines applications commerciales choisissent en conséquence de placer la logique de vérification dans l'application.

Création de « FOREIGN KEYS » dans les tables

- *create table* ***COMMANDES*** *(*
  codecommande int primary key,
  codeclient int UNIQUE,
  FOREIGN KEY (codeclient) REFERENCES CLIENTS,
  codetransporteur int,
  datecommande date, cadeau tinyint, modecommande int,
  nomlivraison char(50), prenomlivraison char(50),
  adresselivraison varchar(255), villelivraison char(50),
  identlivraistran varchar(255), datelivraison datetime,
  modepaiement int UNIQUE,
  FOREIGN KEY (modepaiement) REFERENCES MODE_PAIEMENT;
) type InnoDB;
- ***La FK codeclient* référence la clé primaire de la table *CLIENT***

Lors de toute modification ou insertion, MySQL doit vérifier que la valeur de « codeclient » correspond bien à une ligne de la table CLIENT

Ces vérifications garantissent qu'il n'ya pas de fausses références dans la base (codeclient qui n'existerait pas dans la table CLIENT)

Utilisation des « FOREIGN KEYS » en insertion

```
CREATE TABLE song ( idsong INT NOT NULL ,
        PRIMARY KEY (idsong),
        titre VARCHAR(50),
        duree TIME,
        id_cd INT UNIQUE,
        id_artiste INT UNIQUE,
        FOREIGN KEY (id_cd) REFERENCES disque(idcd),
        FOREIGN KEY (id_artiste) REFERENCES artiste(idartiste)
       ) type = InnoDB;
```

- mysql> insert into song (idsong,titre,duree,id_cd,id_artiste) values (2,"impressions","00:11:20",3,4);
- ERROR 1216: Cannot add or update a child row: a foreign key constraint fails

TP sur Foreign Keys

- Créer une base « disque »
- Création des 3 tables de type InnoDB
  - Artiste
  - Disque
  - chanson
- Modifications de la structure des tables
- Création d'index

Instructions de TRANSACTION

- Le but du mode transactionnel est d'assurer l'intégrité de la base. Un ensemble d'instructions sont exécutées comme si c'était une seule unité de travail.

- En mode transactionnel, si une application a été écrite en dépendant de l'appel de ROLLBACK au lieu de COMMIT dans les situations critiques, les transactions s'assurent que les modifications non achevées ou les activitées corrosives ne sont pas archivées dans la base. Le serveur a l'opportunité d'annuler automatiquement l'opération, et votre base de données est sauve.

Instructions de TRANSACTION

- Introduites récemment dans MySQL avec le format des tables InnoDB
- Par défaut MySQL est en *autocommit* i.e toutes les instructions sont exécutées immédiatement

- Pour passer en mode transactionnel : *mysql> SET AUTOCOMMIT=0;*
- Pour démarrer une transaction : *mysql> BEGIN* ou *START TRANSACTION;*
- *Exécuter un ensemble d'instructions*

  update BOITES set taille=590 where taille=500 ;

  Les modifications ne seront pas permanentes tant qu'on n'aura pas terminé la transaction avec l'instruction COMMIT;
- *Pour TERMINER la transaction (ON REPASSE EN AUTOCOMMIT)*
- *Mysql> COMMIT; (enregistre les modifications de la transaction)*
- *Mysql> ROLLBACK; (annule les modifications de la transaction)*

Instructions de TRANSACTION

- Attention, certaines instructions ne peuvent pas être annulées par des ROLLBACK.

- *Il s'agit des opérations:*

  ALTER TABLE

  CREATE INDEX

  DROP DATABASE

  DROP TABLE

Transactionnel ou pas ???

- Le mode transactionnel n'est pas la panacée. Il est 3 à 4 fois plus lent que le mode non-transactionnel des tables MyISAM
- On peut arriver à de meilleures performances tout en assurant l'intégrité des tables avec le « verrouillage de table » sur des tables MyISAM. La méthode est bien plus rapide que ne le proposent les transactions, avec des annulations ROLLBACK possibles mais pas certaines.

- Les modifications de données transactionnelles fatales peuvent être réécrites de manière « *atomique* ». En général, tous les problèmes d'intégrité que les transactions résolvent peuvent être corrigés avec la commande LOCK TABLES ou des modifications atomiques, qui assurent qu'il n'y aura jamais d'annulation automatique de la base.

"*Atomique*", signifie simplement qu'on peut être certain que lorsqu'on modifie des données dans une table, aucun autre utilisateur ne peut interférer avec votre opération

Transactionnel ou pas?

- *LOCK TABLES table WRITE* pour verrouiller les tables en cours d'utilisation

  Test des conditions , Modification des données si tout est correct.
- *UNLOCK TABLES* pour libérer les tables.

Il existe 3 types de LOCK read, read local, et write

- LOCK TABLES READ : verrouille la table en lecture i.e les écritures ne sont pas possibles
- LOCK TABLES WRITE verrouille la table en lecture et écriture. Les autres clients mysql ne sont autorisés ni à lire ni à écrire

Pendant le verrouillage on est certain que personne ne modifie les données ou ne recueille de fausses informations.

mysql> update BOITES set taille=590 where taille=500 ;

ERROR 1099: Table 'BOITES' was locked with a READ lock and can't be updated

TP sur les transactions

- Engager une transaction SQL
- Insérer des données dans la table CLIENTS
- Vérifier avec commit ou rollback si les données ont été écrites.

Une session PHP-MySQL de base

- mysql_connect() : connexion au serveur mysqld
- mysql_pconnect() : idem avec connexion persistante

  ***$connect***=*mysql_pconnect("localhost::/tmp/mysql.sock",$user,$passwd)*
- mysql_select_db() : selectionne un BD courante

  *mysql_select_db($nombase,**$connect**);*
- mysql_query() : envoie une requête SQL au serveur et récupére le résultat sous forme de tableau associatif ou indicé
  - ***$res***=*mysql_query($query,$connect);*
- mysql_fetch_array() :
  - *$ligne=mysql_fetch_array(**$res**,MYSQL_ASSOC);*
  - *En tableau associatif les champs du tableaux sont indicés par le nom de l'attribut correspondant de la table*

Une session PHP-MySQL de base

- Boucle de lecture : tant qu'il ya des lignes, accéder aux champs du tableau par le nom de l'attribut colonne

```
echo "<li>nom $ligne[nom] ";  echo "prenom $ligne[prenom] ";
```

ou:

```
While ( $ligne=mysql_fetch_array($res,MYSQL_ASSOC) ) {
    if ($ligne) {
        foreach ($ligne as $key => $f) {
            echo "<td align=center>  $f </td>";
        }
}
```

Utilitaires Graphiques de gestion des Bases

- Phpmyadmin:
  - http://www.phpmyadmin.net/
- Mysqlcc :
  - http://dev.mysql.com/downloads/other/mysqlcc.html
- MysqlAdministrator
  - Http://dev.mysql.com/downloads/administrator/