MySQL - Administration

Nathalie Bernier

http://www.linagora.com

Juillet 2002

Table des matières

1 Avant-propos **5**

1.1 Gestion des données 6

1.2 Objectifs d'une base de données 7

1.3 Rôle de l'administrateur de bases de données 9

2 MySQL, un SGBD libre **10**

2.1 Présentation de MySQL 11

2.2 Caractéristiques de MySQL 12

2.3 Installation de MySQL 13

2.3.1 Installation des paquetages 13

2.3.2 Installation des sources 14

2.4 Le client mysql 17

2.4.1 Première connexion 17

2.4.2 Fonctionnement 17

Atelier 2 19

3 Conception et création d'une base de données **20**

3.1 Le modèle relationnel 21

3.1.1 Présentation 21

3.1.2 Les concepts du modèle relationnel 21

3.1.3 L'algèbre relationnelle 23

3.2 La méthode MERISE 27

3.2.1 Les étapes de conception 27

3.2.2 Le Modèle Conceptuel de Données 27

3.2.3 Le Modèle Logique de Données 32

3.3 Le langage SQL 35

3.3.1 Le langage de définition de données 35

3.3.2 Le langage de manipulation de données 39

Atelier 3 44

4 Administration de MySQL **45**

4.1 Présentation du fichier my.cnf 46
4.2 Configuration de MySQL 47
4.2.1 Les options de mysqld 47
4.2.2 Les options de mysql 49
4.2.3 Administration du serveur avec mysqladmin 51
4.3 La gestion des privilèges 52
4.3.1 Introduction 52
4.3.2 Fonctionnement 52
4.3.3 Ajout de privilèges 54
4.3.4 Suppression de privilèges 55
4.3.5 Mots de passe 55
4.3.6 Vérification des privilèges attribués 55
4.4 Sauvegardes et restitutions 57
4.4.1 Sauvegarde avec mysqldump 57
4.4.2 Sauvegarde avec "Select... into outfile ..." 58
4.4.3 Sauvegarde avec la commande tar 59
4.4.4 Restauration 59
Atelier 4 61

5 Optimisation et administration avancée de MySQL **62**

5.1 Les types de tables 63
5.1.1 ISAM 63
5.1.2 MyISAM 64
5.1.3 MERGE 66
5.1.4 HEAP 67
5.1.5 InnoDB 68
5.1.6 BerkeleyDB 73
5.2 La réplication 74
5.2.1 Présentation 74
5.2.2 Mise en oeuvre 75
5.2.3 Les commandes relatives à la réplication 77
5.2.4 Mises en garde : problèmes connus 79
5.3 De la bonne utilisation des index 81
5.3.1 Rôle d'un index 81
5.3.2 Les différents types d'index 81

5.3.3 Quand créer un index ? 82
Atelier 5 83

6 Outils d'administration graphiques 84
6.1 phpMyAdmin : aministration via HTTP 85
6.1.1 Installation 85
6.1.2 Connexion 86
6.1.3 Gestion des utilisateurs 87
6.1.4 Gestion du contenu 88
6.2 MySQL Control Center 90
6.2.1 Présentation 90
6.2.2 Installation 90
6.2.3 Connexion 90
6.2.4 Administration du serveur 91
6.2.5 Gestion des privilèges 93
Atelier 6 95

1
Avant-propos

1.1 Gestion des données

Les données, telles que les adresses des clients, sont au coeur de l'entreprise. Il est donc important que le système responsable de leur gestion soit très stable et accessible à tout moment. Il est capital, dans un système d'informations, de bien faire la différence entre les données elles-mêmes, et le logiciel permettant de les stocker, que l'on appelle le **Système de Gestion de Bases de Données (SGBD)**. MySQL est un Système de Gestion de Bases de Données. C'est un système rapide et robuste, en qui de nombreuses sociétés font confiance comme Yahoo !, MP3.com, Silicon Graphics, ou Texas Instruments.

1.2 Objectifs d'une base de données

Les objectifs des bases de données sont les suivants :

Indépendance physique La façon dont les donnnées sont définies doit être indépendante des structures de stockages utilisées.

Indépendance logique Un même ensemble de données peut être vu différemment par des utilisateurs différents. Toutes ces visions personnelles des données doivent être intégrées dans une vision globale.

Manipulations des données par des non informaticiens Il faut pouvoir accéder aux données sans savoir programmer ce qui implique l'utilisation de langages "quasi naturels".

Efficacité des accès aux données Ces langages doivent permettre d'obtenir des réponses aux interrogations en un temps "raisonnable". Ils doivent donc être optimisés et, entre autres, il faut un mécanisme permettant de minimiser le nombre d'accès disques. Tout ceci, bien sur, de façon complètement transparente pour l'utilisateur.

Administration centralisée des données Des visions différentes des données (entre autres) se résolvent plus facilement si les données sont administrées de façon centralisée.

Non redondance des données Afin d'éviter les problèmes lors des mises à jour, chaque donnée ne doit être présente qu'une seule fois dans la base.

Cohérence des données Les données sont soumises à un certain nombre de contraintes d'intégrité qui définissent un état cohérent de la base. Elles doivent pouvoir être exprimées simplement et vérifiées automatiquement à chaque insertion, modification ou suppression des données.

Partageabilité des données Il s'agit de permettre à plusieurs utilisateurs d'accéder aux mêmes données au même moment. Si ce problème est simple à résoudre quand il s'agit uniquement d'interrogations et quand on est dans un contexte mono-utilisateur, cela n'est plus le cas quand il s'agit de modifications dans un contexte muti-utilisateurs. Il s'agit alors de pouvoir :

- permettre à deux (ou plus) utilisateurs de modifier la même donnée "en même temps" ;
- assurer un résultat d'interrogation cohérent pour un utilisateur consultant une table pendant qu'un autre la modifie.

Sécurité des données Les données doivent pouvoir être protégées contre les accès non autorisés. Pour cela, il faut pouvoir associer à chaque utilisateur des droits d'accès aux données.

Résistance aux pannes Que se passe-t-il si une panne survient au milieu d'une modification, si certains fichiers contenant les données deviennent illisibles ? Bien que robuste et extrêmement testé, un SGBD n'est jamais à l'abris de problèmes, matériels ou logiciels. Il faut pouvoir, lorsqu'un incident se produit, récupérer une base dans un état "sain". Ainsi, après une panne intervenant au milieu d'une modification deux solutions sont possibles : soit récupérer les données dans l'état dans lequel elles étaient avant la modification, soit terminer

l'opération interrompue. Malheureusement, ces objectifs ne sont pas toujours atteints.

1.3 Rôle de l'administrateur de bases de données

L'administrateur de bases de données a en charge :

- La gestion et le support de la base de données.
- L'évaluation des performances d'accès.
- L'évaluation des besoins.
- Les améliorations requises du système.

2

MySQL, un SGBD libre

Objectifs

Savoir installer un serveur MySQL. Première connexion en ligne de commandes.

Contenu

2.1 Présentation de MySQL 11
2.2 Caractéristiques de MySQL 12
2.3 Installation de MySQL 13
2.4 Le client mysql 17
Atelier 2 19

Références

- Le manuel de référence de MySQL situé à l'adresse : http://www.mysql.com/documentation/mysql/bychapter/

2.1 Présentation de MySQL

MySQL est un système de gestion de bases de données relationnelles, compatible avec le langage SQL, édité par la société MySQL AB. Ce système est un logiciel Open Source respectant la licence GPL (GNU Public Licence).
Toutes les informations concernant le(s) serveur(s), les clients, les outils, les développements en cours sont disponibles sur le site officiel : http://www.mysql.com

2.2 Caractéristiques de MySQL

Les caractéristiques principales de MySQL sont ses performances, sa fiabilité et sa facilité d'usage qui en font un acteur majeur des systèmes de bases de données Web. Il présente les caractéristiques suivantes :

Multilangage : mysql peut être interfacé avec la majorité des langages comme le C, C++, Python, Perl, Java, PHP, Tcl.

Multiplate-forme : Windows, Linux, Unix, MacOS X.

Types de colonnes : de nombreux types de colonnes sont disponibles : entiers signés/non signés sur 1, 2, 3, 4, et 8 octets de long, FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET, et ENUM.

Sécurité : l'authentification est basée sur les noms d'utilisateur et d'hôte.

Performances : MySQL peut supporter des bases de données pouvant contenir 5 milliards d'enregistrements.

Connexion : Les connexions au serveur peuvent se faire par socket TCP/IP ou par socket Unix.

2.3 Installation de MySQL

L'installation de MySQL peut être réalisée soit avec des paquetages, soit à partir des sources. Le fait d'installer MySQL à partir des sources nous permet d'adapter davantage le serveur à nos besoins.

2.3.1 Installation des paquetages

Téléchargement

Les paquetages sont généralement disponibles sur les sites des éditeurs de distributions, et sur le site http://www.mysql.com. Les paquetages à télécharger pour une distribution RedHat sont les suivants :

MySQL-shared-XXX.rpm : ce paquetage contient les librairies partagées avec certains langages et certaines applications qui ont besoin de charger des données provenant de MySQL.

MySQL-XXX.rpm : ce paquetage contient la partie spécifique au serveur.

MySQL-Max-XXX.rpm : ce paquetage contient aussi un serveur MySQL, avec des options supplémentaires : support pour les tables InnoDB et Berkeley DB.

MySQL-client-XXX.rpm : ce paquetage contient un client MySQL.

Installation

L'installation en RPM est relativement simple.

Installation du serveur :

```
rpm -Uvh mysql-shared-XXX.rpm mysql-XXX.rpm
```

Installation du client :

```
rpm -Uvh mysql-client-XXX.rpm
```

N.B. : XXX correspond à la version de MySQL.

Démarrage du serveur

Un script *mysql* a été installé dans le répertoire */etc/rc.d/init.d*. Il permet de démarrer le serveur MySQL avec la commande suivante :

```
#/etc/rc.d/init.d/mysql start
```

2.3.2 Installation des sources

Téléchargement

Les sources de MySQL sont disponibles sur le site officiel http://www.mysql.com. Le fichier à télécharger se nomme *mysql-XXX.tar.gz*, ou *XXX* correspond à la version de MySQL. Préférer une version stable à une version en développement.

Décompresser l'archive, et aller dans le dossier obtenu :

```
$tar xvzf mysql-XXX.tar.gz
$cd mysql-XXX
```

Choix des options de compilation

Avant toute chose, s'assurer que le groupe et l'utilisateur *mysql* existent, sinon les créer :

```
$groupadd mysql
$useradd -g mysql mysql
```

Ensuite, le choix des options de compilation se fait à l'aide du script **configure**. Ce script accepte de nombreuses options. Pour en obtenir la liste, utilisez :

```
$./configure --help
```

Voici les principales options :

Option	Explication	Valeur par défaut
--prefix= MYSQL-DIR	MYSQL-DIR est le répertoire d'installation du moteur MySQL	/usr/local
--localstatedir= DATA-DIR	DATA-DIR est le répertoire où installer les fichiers des bases de données. Il est important de vérifier que l'utilisateur Unix qui exécutera MySQL a les droits d'écriture dans ce répertoire	MySQL-DIR/var
--with-mysqld-user= USER	USER est le nom de l'utilisateur Unix qui exécutera le moteur de la base de données	mysql
--enable-local-file	autorise les requêtes "LOAD DATA LOCAL INFILE", c'est-à-dire charger le contenu d'un fichier se trouvant sur la machine cliente dans une table du serveur MySQL	
--with-tcp-port-number= PORT-NUMBER	PORT-NUMBER est le port à utiliser lors des connexions au serveur par TCP	3306
--with-unix-socket-path= SOCKET-PATH	SOCKET-PATH est le chemin vers le fichier socket à utiliser lors des connexions par socket unix	/tmp/mysql.sock
--without-server	ne compile que le client MySQL	
--with-berkeley-db= BK-DIR	BK-DIR est le répertoire où se trouvent les bases de données Berkeley	
--with-innodb	utiliser les bases de données InnoDB	

Exemple : on veut installer MySQL dans */usr/local/mysql*, stocker les bases de données dans */var/mysql* et activer le support *Innodb*. Les options de compilation à choisir sont les suivantes :

```
$./configure  --prefix=/usr/local/mysql --localstatedir=/var/mysql --with-innodb
```

Compilation et installation

La compilation se fait à l'aide de la commande **make** :

```
$make
```

Puis en tant que **root**, exécuter la commande suivante, qui va procéder à l'installation :

```
# make install
```

Initialisation du serveur

Une fois les librairies et les binaires installés, il faut créer les bases par défaut et mettre en place les premiers privilèges. Pour cela on utilise le script `mysql_install_db` présent dans le répertoire `MySQL-DIR/bin` :

```
# cd MySQL-DIR
#./mysql_install_db
```

Cette commande crée :

- les bases de données **mysql** et **test**,
- l'utilisateur **root** sans mot de passe, pouvant uniquement se connecter depuis la machine locale,
- un utilisateur anonyme qui peut se connecter uniquement depuis la machine locale. Il a tous les droits sur les bases dont le nom est "test" ou commence par "test_".

Le serveur étant accessible à tout le monde, une des premières choses à faire est de mettre un **mot de passe** à l'utilisateur **root**, comme expliqué au chapitre 4.3.5.

NB : s'assurer que les droits sont correctement mis en place sur les différents fichiers : les fichiers binaires doivent appartenir à *root*, les fichiers de données doivent appartenir à l'utilisateur qui exécutera le moteur MySQL.

```
#chown -R root  MySQL-DIR
#chown -R mysql MySQL-DIR/var
#chgrp -R mysql MySQL-DIR
```

Démarrage du serveur

On peut désormais tester et démarrer le serveur avec la commande :

```
$ MySQL-DIR/bin/safe_mysqld --user=mysql &
```

Si la commande échoue avec le message `mysqld deamon ended`, on peut trouver des informations dans le fichier *DATA-DIR/hostname.err*, où *hostname* correspond au nom de la machine sur laquelle est installé le serveur MySQL.

Dans le dossier *MySQL-DIR/share/mysql* se trouve le script **mysql.server**. C'est ce script là qu'il faut utiliser pour démarrer le serveur automatiquement. Le copier dans */etc/rc.d/init.d* et faire un lien dans les répertoires correspondant aux niveaux d'exécution, dans lesquels MySQL doit démarrer, vers ce script.

Résumé de l'installation

```
$ groupadd mysql
$ useradd -g mysql mysql
$ gunzip < mysql-VERSION.tar.gz | tar -xvf -
$ cd mysql-VERSION
$ ./configure --prefix=/usr/local/mysql
$ make
$ make install
$ scripts/mysq_install_db
$ chown -R root  /usr/local/mysql
$ chown -R mysql /usr/local/mysql/var
$ chgrp -R mysql /usr/local/mysql
$ cp support-files/my-medium.cnf /etc/my.cnf
$ /usr/local/mysql/bin/safe_mysqld --user=mysql &
$ cp /usr/local/mysql/share/mysql.server /etc/rc.d/init.d
$ ln -s /etc/rc.d/init.d/mysql.server /etc/rc.d/rc3.d
```

Arborescence par défaut

Lors d'une installation à partir des sources, MySQL se trouve par défaut dans le dossier */usr/local*. Les sous-répertoires sont les suivants :

Sous-répertoire	Contenu du sous-répertoire
bin	Programmes et scripts client
include/mysql	Fichiers nécessaires à la compilation d'applications utilisant MySQL
info	Documentation au format Info
lib/mysql	Librairies
libexec	Le serveur 'mysqld'
share/mysql	Fichiers de messages d'erreur
sql-bench	Utilitaires pour tester le serveur MySQL
var	Fichiers de log et bases de données. Dans ce répertoire se trouve un fichier nommé *hostname.err* dans lequel se trouvent tous les messages d'erreurs rencontrées par le serveur

2.4 Le client mysql

Nous allons dans cette partie présenter brièvement le client mysql installé soit par paquetage, soit lors de la compilation des sources. L'exécutable se nomme **mysql** et se trouve dans */usr/bin* pour la version paquetage ou dans *MySQL-DIR*/bin pour la version compilée. C'est un client en ligne de commande : tout se fait en mode shell.

Il est possible de définir des paramètres de connexion par défaut pour le client mysql. Nous verrons lesquelles et comment au chapitre 4.2.2.

2.4.1 Première connexion

A ce niveau de l'installation, seul l'utilisateur **root** existe, et il n'a pas de mot de passe, il faut donc taper la commande ci-dessous pour se connecter au serveur :

```
shell> mysql -u root
mysql>
```

2.4.2 Fonctionnement

Le client mysql permet d'exécuter des requêtes pour manipuler les bases de données ou maintenir les bases et les tables. Il met également à notre disposition des commandes dont quelques-unes sont expliquées ci-après.

Soumettre des requêtes

Chaque requête se termine par un point-virgule, comme dans l'exemple ci-dessous :

```
mysql> SELECT VERSION(), CURRENT_DATE;
+--------------+--------------+
| VERSION()    | CURRENT_DATE |
+--------------+--------------+
| 3.22.20a-log | 1999-03-19   |
+--------------+--------------+
1 row in set (0.01 sec) mysql>
```

Plusieurs requêtes peuvent être exécutées sur la même ligne de commnande :

```
mysql> SELECT VERSION();select CURRENT_DATE;
+--------------+
| VERSION()    |
+--------------+
| 3.22.20a-log |
+--------------+
1 row in set (0.00 sec)
```

```
+----------------+
| CURRENT_DATE() |
+----------------+
| 1999-03-19     |
+----------------+
1 row in set (0.03 sec)

mysql>
```

Les commandes

Taper `help` pour avoir la liste complète. A noter les commandes suivantes :

exit, quit quitter mysql.

use utiliser une autre base de données. Attend le nom de la base de données en paramètre.

status obtenir les informations du serveur.

rehash activer la complétion sur les noms de bases, tables, champs et commandes.

source exécute un script SQL se trouvant dans le fichier passé en paramètre.

Exécution de script

Comme on l'a vu précédemment, un script SQL peut être exécuté grâce à la commande `source` dans le shell mysql :

```
mysql>use nom_base
mysql>source script.sql
```

Mais il peut aussi être exécuté dans le shell Linux de cette manière :

```
shell>mysql nom_base < script.sql > resultat.tab
```

Dans l'exemple ci-dessus, les résultats sont redirigés dans le fichier `resultat.tab`.

Atelier 2 : MySQL, un SGBD libre

Exercice 2.1 *Installation de MySQL à partir des sources :*

1. Téléchargez l'archive sur le site http ://www.mysql.com.
2. Compilez MySQL avec le support InnoDB.
3. Lancez le serveur MySQL, et faites en sorte qu'il le soit automatiquement au prochain démarrage de la machine. Rebootez et vérifiez

Exercice 2.2 *Première connexion*

Connectez-vous au serveur à l'aide du client texte de mysql, et tapez la commande **status**. Comment êtes-vous connecté ?

3

Conception et création d'une base de données

Objectifs

- Comprendre les concepts du modèle relationnel.
- Savoir concevoir une base de données.
- Connaître les bases du langage SQL.

Contenu

3.1 Le modèle relationnel 21
3.2 La méthode MERISE 27
3.3 Le langage SQL 35
Atelier 3 44

Références

- Introduction à la conception de bases de données relationnelles, de Didier Boulle. http://www.iut.univ-st-etienne.fr/coursgea/informatique/introbd/index.htm
- Introduction au langage SQL : http://www.commentcamarche.com/sql/sqlintro.php3

3.1 Le modèle relationnel

Les concepts à connaître avant de concevoir une base de données.

3.1.1 Présentation

Le modèle relationnel a été formalisé par CODD qui a défini 12 règles entre 1970 et 1975. Pour ce faire, il s'est inspiré de la théorie mathématique des ensembles.

Les objectifs du modèle relationnel sont les suivants :

- Permettre un haut degré d'indépendance entre les applications (programmes, interfaces) et la représentation interne des données (fichiers, chemins d'accès)
- Etablir une base solide pour traiter les problèmes de cohérence et de redondance des données.

Le principe du modèle relationnel est de représenter les données sous forme de tables(tableaux de valeurs) où chaque table représente une relation au sens mathématique d'**Ensemble**.

3.1.2 Les concepts du modèle relationnel

Pour introduire le modèle relationnel, nous allons nous appuyer sur un exemple concret : la gestion de prêt de disques dans une médiathèque. Comme nous venons de le voir dans le paragraphe précédent, une base de données est composée de tables. La base de données en question comporte les tables suivantes :

DISQUE :

CodeDisc	Titre	Auteur
1	Boucan d'enfer	Renaud
2	Californication	Red Hot Chili Peppers
3	Yellow Submarine	The Beatles
4	Heathen	David Bowie

EXEMPLAIRE

CodeDisc	NumEx	DateAchat	Etat
1	1	2002-07-24	neuf
1	2	2002-07-24	neuf
2	1	2002-01-12	bon
2	2	2002-07-24	neuf
3	1	2001-07-24	bon

ABONNE

NumAbon	Nom	Prenom	Adresse	Telephone
1	Dupont	Jean	4 Grande Rue - 75002 PARIS	0383146474
2	Cartier	Mathieu	5 Rue de Lilas - 75003 PARIS	0125484753
3	Michel	Martine	49 Bd Voltaire - 75011 PARIS	0124478543

EMPRUNT

CodeDisc	NumEx	Date	NumAbon
2	2	2002-07-23	2
1	1	2002-08-12	3
1	2	2002-08-16	2
3	1	2002-08-16	3

Les notions à introduire dans ce paragraphe font référence au modèle relationnel, et à l'algèbre relationnelle.

Définitions

Attribut : chaque colonne d'une table est un attribut.

Enregistrement : chaque **ligne** d'une table est un enregistrement, encore appelé **n-uplet**, ou **n** représente le nombre d'attributs de la relation.

Domaine : un domaine D est un ensemble de valeurs caractérisé par un nom. Du point de vue du modèle relationnel, chaque valeur du domaine est atomique et donc indivisible. Cette notion permet de définir les ensembles de départ. Un domaine peut être défini en extension en donnant la liste des valeurs composantes ou en compréhension en définissant une propriété caractéristique du domaine.
Du point de vue de la réalisation informatique, le domaine se restreint à la notion de type de données. Néanmoins, il est essentiel au cours de l'étape de conception de clairement définir les domaines.
Exemple : dans la table DISQUE, la colonne **CodeDisc** sera de type **entier**, **entier** est alors le domaine de la colonne CodeDisc.

Relation : Une relation **r** dénotée **r(R)** du schéma de relation **R(A1 :D1, A2 :D2, ..., An :Dn)** est un ensemble d'enregistrements. Chaque enregistrement **ei** est une liste ordonnée de **n** valeurs **ei** $=<$ **v1**, **v2**, ..., **vn** $>$ ou chaque **vi** est une valeur du domaine de l'attribut **Ai** ou une valeur nulle spéciale représentant l'absence d'information.
Exemple : *DISQUE, ABONNE, EXEMPLAIRE, et EMPRUNT* sont quatre relations.

Clé d'une relation : une clé de relation est un **sous-ensemble d'attributs** qui permet de caractériser tout enregistrement d'une relation. Par définition, une relation est un ensemble d'enregistrements et il ne peut donc pas y avoir deux

enregistrements strictement identiques dans la même relation. De manière informelle, une clé est un ensemble minimum d'attributs dont la connaissance des valeurs permet d'identifier un enregistrement unique de la relation considérée. Une clé est invariante dans le temps. En général, il existe plusieurs clés pour une même relation R. Parmi les clés possibles, on choisit une clé qui sera appelée clé primaire . Lors de la définition d'un schéma cette clé est mise en évidence (soulignement).
Exemple : l'attribut **NumAbon** est la clé primaire de la relation ABONNE.

Schéma de base de données et contraintes d'intégrité : un schéma de base de données relationnelles S est un ensemble de schémas de relations S = { R1, R2, ..., Rp } et un ensemble de contraintes d'intégrité CI.

Une contrainte d'intégrité est une propriété du schéma, invariante dans le temps. Il existe différents types de contraintes d'intégrité :
- contraintes liées au modèle (pas de doublons dans une relation.) ;
- contraintes de domaine (nb_heure < 100 ; pas de valeur nulle pour la clé primaire) ;
- contraintes référentielles dites de clé étrangère qui impose que la valeur d'attribut de la relation r1 apparaîssent comme valeur de clé dans une autre relation r2.

Schéma de relation : un schéma de relation R, dénoté R(A1 :D1, A2 :D2, ..., An :Dn) est un ensemble d'attributs. Chaque attribut Ai est le nom d'un rôle joué par son domaine Di dans le schéma de relation R.
Un schéma de relation R est utilisé pour décrire une relation.
Exemple : le schéma relationnel précède la réalisation de la base de données. Il permet définir de manière précise les attributs composant chaque table. Le schéma de relation correspondant à la base ci-dessus est le suivant :
- DISQUE (CodeDisc, Titre, Date, Auteur)
- EXEMPLAIRE (CodeDisc, NumEx, DateAchat, Etat)
- ABONNE (NumAbo, Nom, Prénom, Adresse, Téléphone)
- EMPRUNT (CodeDisc, NumEx,DatePrêt,#NumAbo)

Dans un schéma relationnel, ce qui sera plus tard une table dans la base de données est représenté par le nom de la "table" suivi entre parenthèses des différents attributs la composant. Pour identifier les différentes clés : les clés primaires sont soulignées et les clés étrangères sont suivies d'un #. Une clé étrangère est la clé primaire d'une autre table.

3.1.3 L'algèbre relationnelle

L'algèbre relationnelle est une collection d'opérations permettant d'opérer sur les concepts du modèle relationnel. Elle permet par exemple de sélectionner certains enregistrements d'une relation satisfaisant une condition ou encore de regrouper des enregistrements de relations différentes.

Le résultat de toute opération de l'algèbre est une nouvelle relation. Cette propriété implique notamment qu'il n'y a pas de doublons dans le résultat et permet l'écriture d'expressions de calcul.

Etant basée sur la théorie des ensembles, l'algèbre relationnelle utilise les opérateurs classiques de manipulation des ensembles (union, intersection, différence et produit cartésien) et introduit des opérateurs propres aux bases de données (sélection, projection, jointure, division).

Les opérateurs ensemblistes

Union compatible : deux relations sont union-compatibles si elles ont le même nombre d'attributs et que ceux-ci ont le même domaine. Les opérateurs union ∪ et intersection ∩ nécessitent que les relations soient union-compatibles.

Opérateur union ∪ : soient R et S, deux relations de schémas respectifs X et Y. Les schémas X et Y doivent être union-compatibles c'est à dire posséder le même nombre d'attributs et que ceux-ci soient de même domaine. L'union des deux relations, R ∪ S produit une nouvelle relation de schéma identique à R possédant les enregistrements appartenant à R ou à S ou aux deux relations.
Exemple : soient les deux relations ci-dessous :
La liste de tous les exemplaires du disque dont le CodeDisc=1 et etat=neuf :

CodeDisc	NumEx	DateAchat	Etat
1	1	2002-07-24	neuf
1	2	2002-07-24	neuf

La liste de tous les exemplaires du disque dont le CodeDisc=2 et etat=neuf :

CodeDisc	NumEx	DateAchat	Etat
2	2	2002-07-24	neuf

La liste de tous les exemplaires des disques numéro 1 et 2 étant neuf est l'union ci-dessous :

CodeDisc	NumEx	DateAchat	Etat
1	1	2002-07-24	neuf
1	2	2002-07-24	neuf
2	2	2002-07-24	neuf

Opérateur intersection ∩ : soient R et S, deux relations de schémas respectifs X et Y. Les schémas X et Y doivent être union-compatibles. L'intersection des deux relations R ∩ S produit une nouvelle relation de schéma identique à R possédant les enregistrements appartenant conjointement à R et à S.
Exemple : soient les deux relations suivantes :
La liste de tous les exemplaires des disques 1 et 2 neufs :

CodeDisc	NumEx	DateAchat	Etat
1	1	2002-07-24	neuf
1	2	2002-07-24	neuf
2	2	2002-07-24	neuf

La liste des disques empruntés par Martine :

CodeDisc	NumEx	Date	NumAbon
1	1	2002-08-12	3
3	1	2002-08-16	3

L'intersection de ces deux relations correspond à tous les disques neufs empruntés par Martine :

CodeDisc	NumEx	Date	NumAbon
1	1	2002-08-12	3

Opérateur différence - : soient R et S, deux relations de schémas respectifs X et Y. Les schémas X et Y doivent être union-compatibles. La différence des deux relations R - S produit une nouvelle relation de schéma identique à R possédant les enregistrements présents dans R mais pas dans S.

Opérateur produit cartésien x : le produit cartésien est un opérateur issu de la théorie des ensembles défini comme suit : si A et B sont deux ensembles, leur produit cartésien **A X B** contient toutes les paires (a, b) avec a ∈ A et b ∈ B. Ceci signifie que le produit cartésien permet d'obtenir toutes les combinaisons possibles entre les éléments de deux ensembles. Dans le cadre de l'algèbre relationnelle nous définirons donc le produit cartésien comme suit :

Soient R et S, deux relations de schémas respectifs X et Y. Les schémas X et Y doivent être disjoints c'est-à-dire ne pas avoir d'attributs communs. Le produit cartésien des deux relations R X S produit une nouvelle relation de schéma Z égal à l'union des schémas R et S et possédant comme enregistrements, la concaténation des enregistrements de R avec ceux de S.

Les opérateurs de bases de données

Sélection : sélection prend en entrée une relation R de schéma X et produit en sortie une nouvelle relation de schéma X ayant comme enregistrements ceux de R satisfaisant la condition de sélection. La condition de sélection utilise les opérateurs de comparaison ($<; <=; >; =>; =; ! =$), les connecteurs logiques (et, ou, non) et les parenthèses.

Exemple : la relation suivante correspond à la sélection de tous les disques achetés le 24 juillet 2002 :

CodeDisc	NumEx	DateAchat	Etat
1	1	2002-07-24	neuf
1	2	2002-07-24	neuf
2	2	2002-07-24	neuf

Projection ⊆ : la projection prend en entrée une relation R de schéma X et produit en sortie une nouvelle relation de schéma `A1 ; A2 ; ... ; An` (schéma inclu dans X) ayant comme enregistrements ceux de R restreints au sous-schéma `A1 ; A2 ; ... ; An`, c'est-à-dire sélectionner uniquement certains champs d'une table (ou relation)

Exemple : si on considère la relation précédente, nous listant les disques

achetés depuis le 24 juillet 2002, on peut ne demander en résultat que les colonnes correspondant au numéro d'exemplaire et au code du disque. Ainsi on obtiendrait :

CodeDisc	NumEx
1	1
1	2
2	2

Produit ou jointure $\bowtie$: Soient R et S deux relations de schéma respectif X et Y. Il faut que les schémas X et Y possèdent une intersection Z non vide. La relation résultat a un schéma qui est l'union des deux schémas X et Y et comme enregistrements la concaténation des enregistrements de R avec ceux de S s'ils ont la même valeur pour les attributs communs.

Exemple : de manière à rendre le résultat précédent plus lisible par l'utilisateur, on aimerait avoir en plus des codes des disques et du numéro d'exemplaire, le titre et l'auteur correspondant. La jointure se fait alors grâce à la colonne CodeDisc, commune aux tables DISQUE et EXEMPLAIRE :

CodeDisc	NumEx	Titre	Auteur
1	1	Boucan d'enfer	Renaud
1	2	Boucan d'enfer	Renaud
2	2	Californication	Red Hot Chili Peppers

Division $\div$: Soient les relations R(X,Y) et S(X). **T** est la relation résultat de la division de R par S. Elle est définie sur le schéma X et la concaténation de tous ses enregistrements avec ceux de S appartiennent à R.

Syntaxe : $R(X,Y) \div S(Y) = T(X)$

3.2 La méthode MERISE

Lors de la conception d'un système d'informations, les questions à se poser sont les suivantes :

- *Comment identifier les données (variables, constantes,...) ?*
- *Comment structurer les données (liste, table, arbre ...) ?*
- *Comment les données interagissent ?*

3.2.1 Les étapes de conception

La méthode MERISE, terminée en 1979, intervient dans la phase de conception de la base de données. Elle se découpe en quatre étapes :

- Etude de l'existant
- Réalisation du modèle conceptuel de données (MCD)
- Réalisation du modèle logique de données (MLD)
- Mise en oeuvre du modèle physique de données (MPD)

3.2.2 Le Modèle Conceptuel de Données

L'objectif du MCD est de proposer une représentation schématique des données de l'entreprise en utilisant le formalisme d'entité/association qui est abordable par tout utilisateur et permet un dialogue clair avec les concepteurs. Le MCD obtenu ne doit inclure que des données nécessaires pour le fonctionnement de l'entreprise et les liens existants entre ces données.

La réalisation d'un MCD se fait en plusieurs étapes : la liste toutes les données et leur type (entier, chaîne de caractères ...), puis la création du graphe des dépendances fonctionnelles que l'on transforme enfin en MCD.

Le dictionnaire de données

Exemple : L'exemple ci-dessous sera utilisé tout au long de la présentation de la méthode MERISE.

Soit la liste des données recensées dans un établissement scolaire :

- numéro identifiant de l'élève, un entier non signé qui s'autoincrémente,
- nom de l'élève, chaîne de 50 caractères maximum,
- prénom de l'élève, chaîne de 50 caractères maximum,
- adresse de l'élève,chaîne de 100 caractères maximum,
- numéro identifiant de la matière,un entier non signé qui s'autoincrémente,
- matière enseignée, chaîne de 100 caractères maximum,
- nom du professeur, chaîne de 50 caractères maximum,
- nombre d'heures, entier non signé,

- numéro identifiant de la classe,un entier non signé qui s'autoincrémente,
- nom de la classe, chaîne de 10 caractères maximum,
- numéro de la salle,entier non signé,
- note, entier non signé compris entre 0 et 20 inclus.

Soient les règles de gestion suivantes :

- A chaque classe est attribuée une et une seule salle,
- Chaque matière est enseignée par un et un seul professeur, et un professeur n'enseigne qu'une matière.
- Pour chaque classe et chaque matière est défini un nombre fixe d'heures de cours,
- A chaque élève est attribuée un seule note par matière,
- L'établissement gère les emplois du temps des professeurs et des élèves ainsi que le contrôle des connaissances.

Le graphe des dépendances fonctionnelles

Il s'agit là de définir la dépendance existant entre les différentes données du dictionnaire. Les dépendances entre les données de l'exemple ci-dessus sont les suivantes :

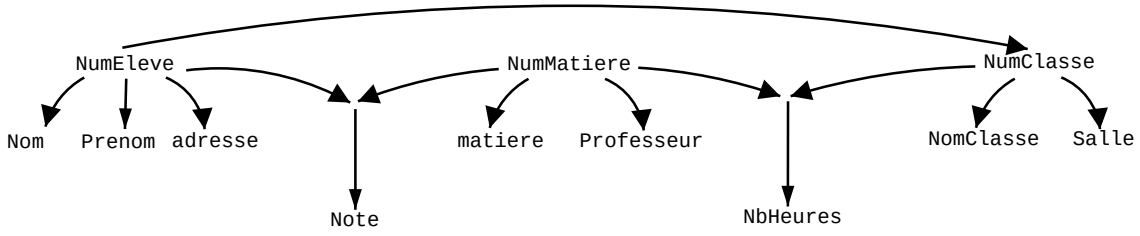


FIG. 3.1: *Graphe des dépendances fonctionnelles*

Transformation du GDF en MCD

Les données étant originaires d'au moins une dépendance fonctionnelle correspondent aux identifiants des entités du MCD, c'est le cas de **NumEleve, NumMatiere, et NumClasse**. Nous aurons donc trois entités : **Eleve, Matiere et Classe**.

Les données qui dépendent de plus d'un objet décrivent les associations entre ces objets, et on obtient les relations *avoir pour note* et *suivre pendant*, ayant respectivement pour attribut Note et NbHeures.

- "Eleve" "avoir pour note" "matière"
- "Classe" "suivre pendant" "matière"

Représentation des entités/associations

Les entités : les objets indépendants sont des entités. Nous aurons donc les

entités **classe**, **élève** et **matière**. Elles sont représentées dans le MCD sous forme de rectangle divisé en deux parties : le nom de l'entité et la liste des attributs qui la composent. Exemple :

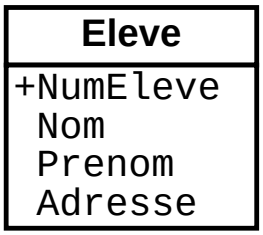


FIG. 3.2: *Entité Elève*

Les associations : c'est la représentation des liens entre chaque objet. Une association se présente sous forme d'ovale dans le MCD, composé de deux parties : le nom de l'association et la liste des attributs qu'elle comporte.

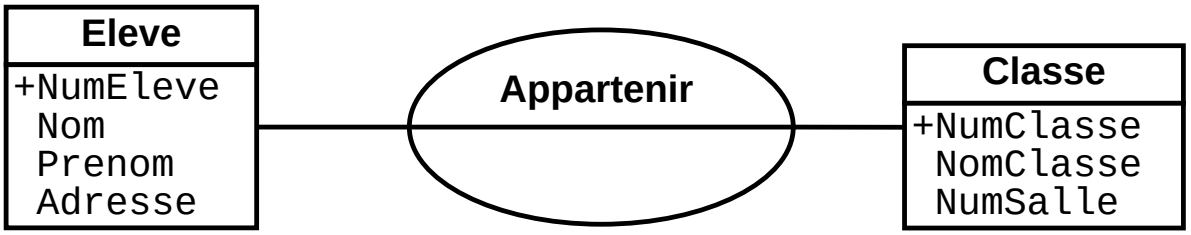


FIG. 3.3: *Association*

La dimension d'une association : le nombre d'occurences des entités concernées par l'association est appelé sa dimension.

Exemple 1 : Dimension=2, association binaire

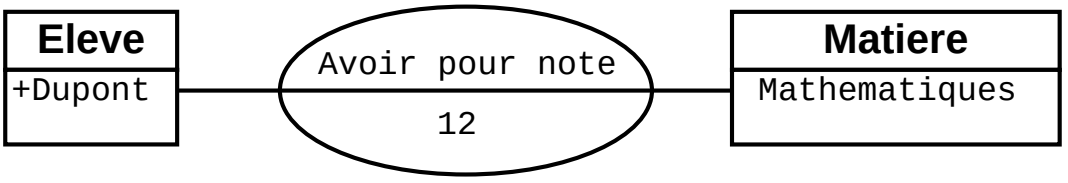


FIG. 3.4: *Association binaire*

Exemple 2 : Dimension=3, association ternaire

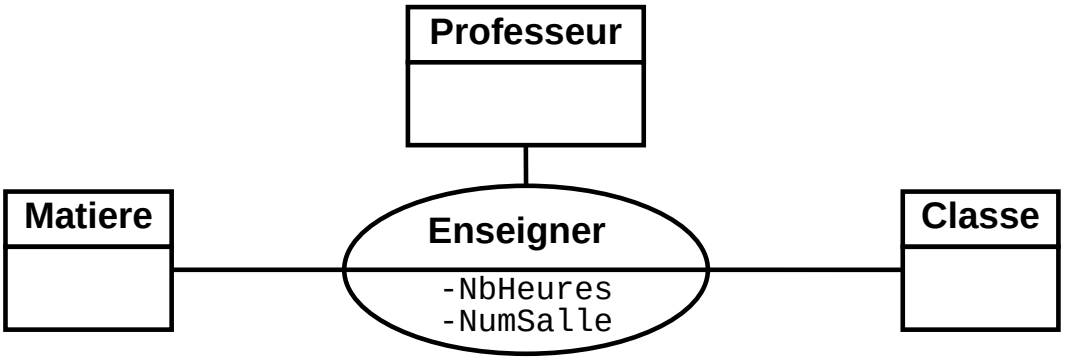
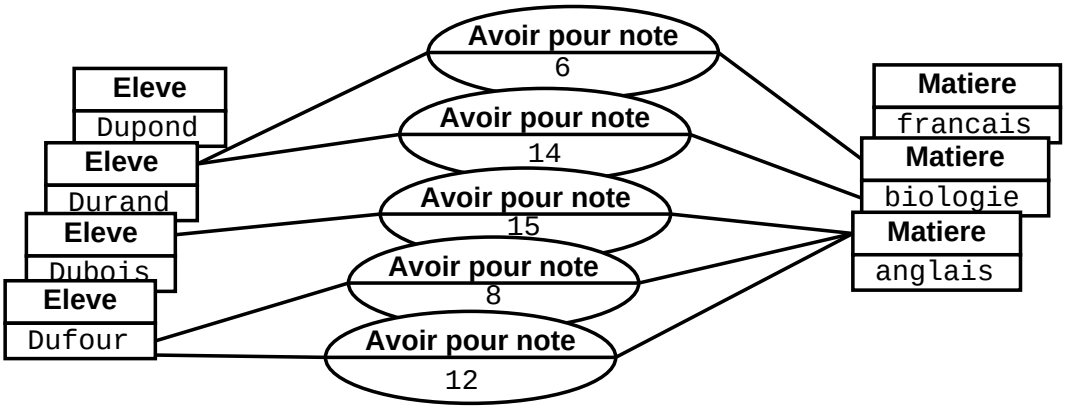


FIG. 3.5: *Association ternaire*

Mise en place des cardinalités

Les cardinalités d'une entité dans une association mesurent, lorsque l'on parcourt l'ensemble des occurences de cette entité, le minimum et le maximum de leur participation à l'association. Sur le MCD, les cardinalités sont représentées comme suit : **(min,max)**.

Exemple :



La participation des occurences **élève** est comprise entre 0 et 2 : *Dupond* n'a pas de note,*Dufour et Durand* en ont deux. La cardinalité de "élève" est donc **(0,2)**
La participation de **matière** est comprise entre 0 et 3 : *français* ne participe pas du tout à l'association, **anglais** y participe trois fois. La cardinalité de "matière" est donc **(0,3)**

En règle générale, on utilise n pour remplacer une cardinalité maximale supérieure à 1. On a alors quatre types de cardinalités :

(0,1) : une occurence de l'entité ne participe jamais plus d'une fois à l'association.

(1,1) : une occurence de l'entité participe toujours une et une seule fois à l'association.

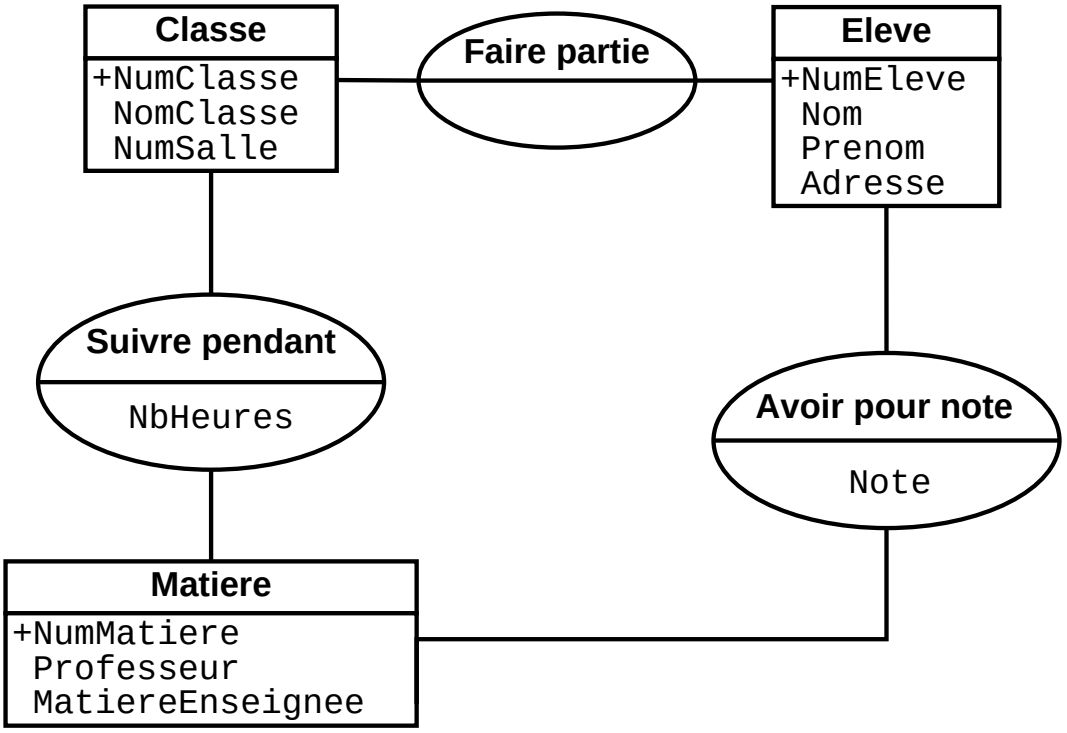
(1,n) : une occurence de l'entité participe toujours au moins une fois à l'association.

(0,n) : aucune précision quant à la participation des occurences de l'entité à l'association.

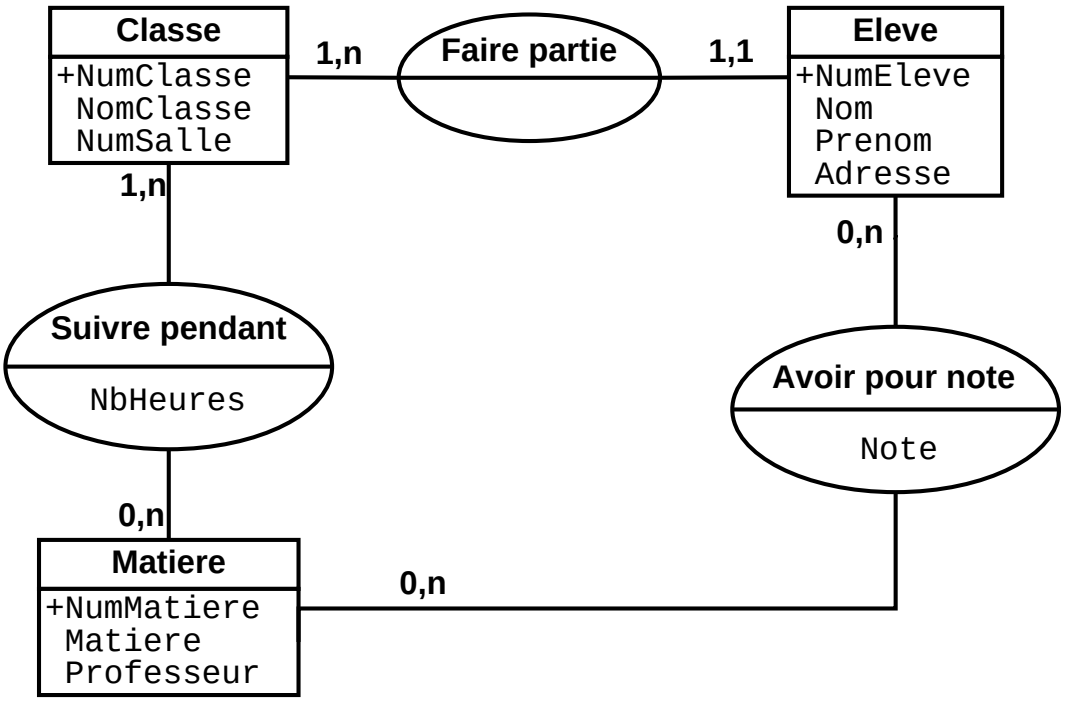
Réalisation du MCD

Les règles de gestion précisent les contraintes qui doivent être respectées par le modèle.

Exemple :



Les cardinalités sont définies de cette manière :



En effet, un élève fait obligatoirement partie d'une et une seule classe (d'où la cardinalité **(1,1)**), tandis que dans une classe on aura entre un et plusieurs élèves (cardinalité **(1,n)**).

De même un élève aura entre 0 et plusieurs notes (toutes matières confondues), et pour une même matière, plusieurs notes seront attribuées.

3.2.3 Le Modèle Logique de Données

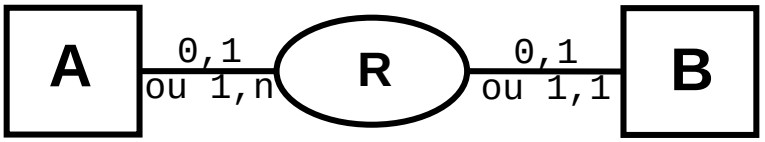
Présentation

Le modèle logique de données est la dernière étape avant la création de la base de données. Il permet de définir exactement tous les attributs de toutes les tables, ainsi que les contraintes d'intégrité.

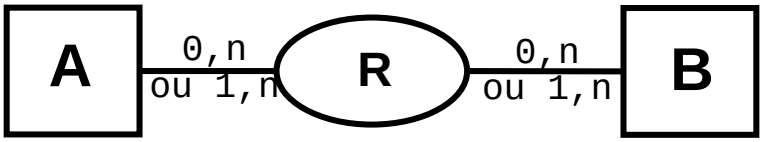
Règles de transformation du MCD en MLD

- Les entités du MCD deviennent des relations au sens relationnel(c'est-à-dire les tables).
- Leurs attributs deviennent des constituants (l'identifiant devenant une clé primaire) des tables.
- Quant aux associations, on distingue trois cas de liaisons binaires entre entités :

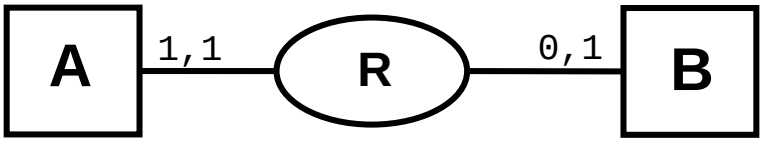
  1-n : R disparaît, l'identifiant de A étant incorporé à la relation B (au sens relationnel, c'est-à-dire table B) en tant que clé étrangère. Si R est porteuse d'attributs, ceux-ci deviennent des constituants de B.



n-n : R devient une relation au sens relationnel(table), sa clé primaire étant obtenue en concaténant les identifiants des entités qui participent à cette R. Les attributs portés par R deviennent les constituants de cette nouvelle table.



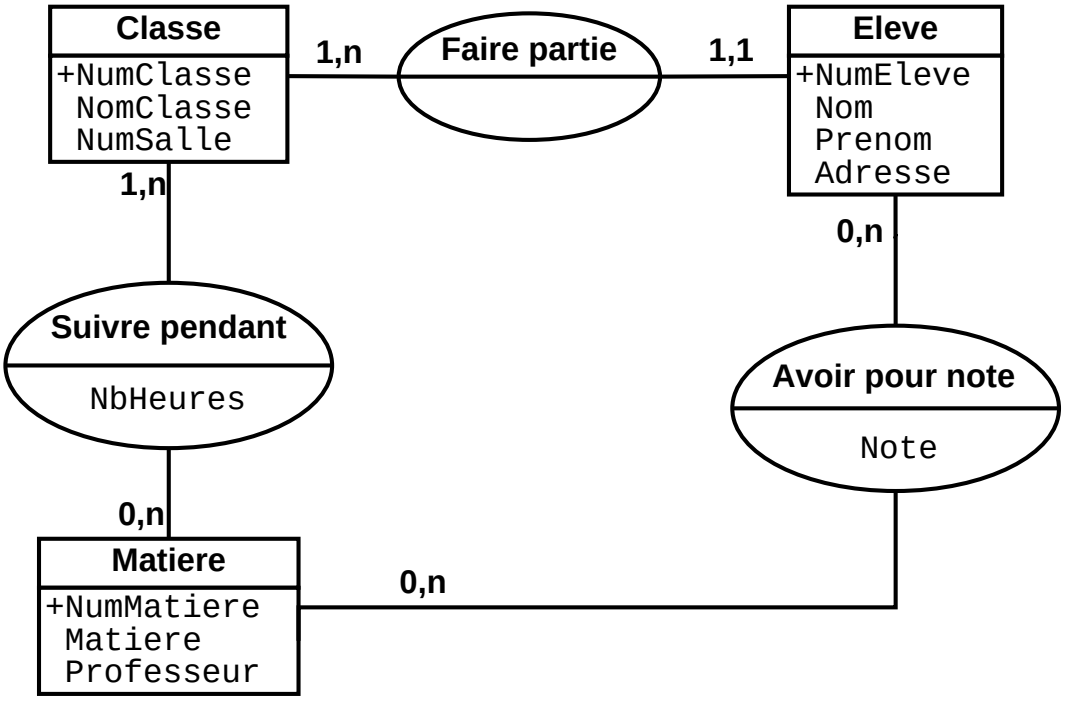
1-1 : R disparaît, comme dans le cas 1-n. L'entité émettrice de la dépendance fonctionnelle A reçoit l'identifiant de B en tant que clé étrangère.



On obtient le type de liaison en prenant le maximum des cardinalités de chaque entité participant à la relation.

Transformation du MCD en MLD

Le MCD obtenu précédemment :



nous donne le MLD suivant :

- **CLASSE**(NumClasse, NomClasse, NumSalle)
- **MATIERE**(NumMatiere, MatiereEnseignee, Professeur)
- **ELEVE**(NumEleve,#NumClasse, Nom, Prenom, Adresse)
- **NOTE**(#NumEleve, #NumMatiere, Note)
- **HORAIRE**(#NumClasse,#NumMatiere, NbHeures)

Chaque relation du MLD représente soit une entité du MCD, soit une association de type n-n. Dans le MCD, l'association entre l'entité ELEVE et l'entité CLASSE étant de type 1-n, l'identifiant de l'entité CLASSE devient une clé étrangère de la table ELEVE dans le MLD. Lors de la création de la base de données, nous aurons donc 5 tables à créer : CLASSE, MATIERE, ELEVE, NOTE, et HORAIRE.

Optimisation du MLD

Le MLD construit précédemment est conceptuellement optimal : tous les liens ont été pris en compte, il n'y a pas de redondance de données ...

Mais on doit néanmoins prendre en considération le niveau opérationnel : quel SGBD est utilisé, quelle machine ... dans le but de minimiser le temps d'accès aux enregistrements. Pour optimiser le temps d'accès aux données, on créera des index. Voir au chapitre 5.3.

3.3 Le langage SQL

Comment mettre en oeuvre le MLD défini lors de la conception de la base de données ? Comment manipuler les données ? Comment réaliser un MPD ?

SQL (Structured Query Language, traduisez Langage de requêtes structuré) est un langage de définition de données (LDD, ou en anglais DDL Data Definition Language) et un langage de manipulation de données (LMD, ou en anglais DML, Data Manipulation Language) pour les bases de données relationnelles.

Après l'invention du modèle relationnel par E.F. Codd, de nombreux langages ont fait leur apparition :

- IBM Sequel (Structured English Query Language) en 1977
- IBM Sequel/2
- IBM System/R
- IBM DB2

Ce sont ces langages qui ont donné naissance au standard SQL.

3.3.1 Le langage de définition de données

SQL est un langage de définition de données (LDD), c'est-à-dire qu'il permet de créer des tables dans une base de données relationnelle, ainsi que d'en modifier ou d'en supprimer.

Création d'une base de données

Sous MySQL, on considère une base de données comme un conteneur, à l'intérieur de laquelle on trouvera les tables.

La création d'une base de données s'effectue avec la commande **CREATE DATABASE** :

```
mysql>CREATE DATABASE [IF NOT EXISTS] nom_base
```

Création d'une table

Le création de tables se fait à l'aide du couple de mots-clés **CREATE TABLE**. La syntaxe simplifiée de définition d'une table est la suivante :

```
mysql>CREATE  TABLE [IF NOT EXISTS] nom_table
[(nom_colonne1 type_donnees,nom_colonne2 type_donnees...)]
```

Le choix du type de données est très important, il faut être sûr, par exemple, que toutes les informations qui devront être stockées dans chaque colonne ne dépasseront pas la longueur prévue pour chacune d'entre elles. Le tableau ci-dessous propose quelques types disponibles sous MySQL :

Type de donnée	Syntaxe	Description
alphanumérique	CHAR(n)	Chaîne de caractères de longueur fixe n (1<n<255)
alphanumérique	VARCHAR(n)	Chaîne de caractères de longueur variable, où n est le nombre de caractères maximum (1<n<255)
texte	TEXT	Texte (Chaîne de caractères) de longueur variable, non sensible à la case
blob	BLOB	Texte (Chaîne de caractères) de longueur variable sensible à la case
numérique	NUMBER(n,d)	Nombre de n chiffres avant la virgule et d chiffres après.
numérique	INTEGER Entier	signé de 32 bits (-2E31 à 2E31-1)
numérique	SMALLINT	Entier signé de 16 bits (-32768 à 32757)
numérique	FLOAT	Nombre à virgule flottante
date	DATE	Date sous la forme 'JJ-MM-AA'
horaire	TIME	Heure sous la forme 'HH :MM :SS'
date et heure	DATETIME	Date et Heure sous la forme "JJ-MM-AA HH :MM :SS'
date et heure	TIMESTAMP(n)	Date et Heure au format AAAAMMJJHHMMSS. Sa longueur dépend de n : TIMESTAMP(14) est au format 'AAAAMMJJHHMMSS', TIMESTAMP(12) est au format 'AAMMJJHHMMSS', TIMESTAMP(10) est au format 'AAMMJJHHMM', TIMESTAMP(8) est au format 'AAAAMMJJ'

Contraintes d'intégrité

Une contrainte d'intégrité est une clause permettant de contraindre la modification de tables, faite par l'intermédiaire de requêtes d'utilisateurs, afin que les données saisies dans la base soient conformes aux données attendues. Ces contraintes doivent être exprimées dès la création de la table grâce aux mots-clés suivants :

DEFAULT : donne la valeur par défaut d'une colonne.

NOT NULL : la colonne ne pourra pas prendre la valeur NULL. On l'utilise souvent dans les colonnes étant des clés primaires. Si aucune valeur n'est fournie par la requête d'insertion, une erreur sera retournée, et l'enregistrement ne sera pas fait.

UNIQUE : chaque valeur de la colonne doit être différente de toutes les autres.

AUTO_INCREMENT : cette option est disponible pour les colonnes de type entier. La valeur de chaque enregistrement d'une colonne comportant cette option est égale à la dernière valeur insérée +1. Il ne peut y avoir qu'une colonne qui s'autoincrémente par table et elle doit être indexée.

PRIMARY KEY : définit la clé primaire d'une table.

INDEX : crée un index à la table. Voir le paragraphe 3.3.1

Exemple :

```
mysql> create table CLASSE (
NumClasse int AUTO_INCREMENT NOT NULL PRIMARY KEY,
NomClasse varchar(10),
NumSalle smallint unsigned);
```

Création d'un index

Un index est un objet complémentaire (mais non indispensable) à la base de données permettant d'"indexer" certaines colonnes dans le but d'améliorer l'accès aux données par le SGBDR, au même titre qu'un index dans un livre ne vous est pas indispensable mais vous permet souvent d'économiser du temps lorsque vous recherchez une partie spécifique de ce dernier...

Toutefois la création d'index utilise de l'espace mémoire dans la base de données, et, étant donné qu'il est mis à jour à chaque modification de la table à laquelle il est rattaché, peut alourdir le temps de traitement du SGBDR lors de la saisie de données. Par conséquent il faut que la création d'index soit justifié et que les colonnes sur lesquelles il porte soient judicieusement choisies (de telle façon à minimiser les doublons). De cette façon certains SGBDR créent automatiquement un index lorsqu'une clé primaire est définie.

La syntaxe de création d'index est la suivante :

```
mysql>CREATE INDEX Nom_de_l_index ON Nom_de_la_table(Nom_de_champ)
```

Exemple : Création d'un index sur les 5 premiers caractères du champ *NomClasse* de la table CLASSE :

```
mysql create index index_classe on CLASSE(NomClasse(5))
```

suppression d'éléments

La commande DROP peut supprimer différents types d'éléments d'une base de données. Tout d'abord, elle peut supprimer une base de données :

```
mysql> DROP DATABASE nom_base
```

Elle permet également de supprimer une table :

```
mysql> DROP TABLE nom_table
```

Enfin la suppression d'un index sur une table se fait de cette manière :

```
mysql>DROP INDEX index_name ON tbl_name
```

Modification de la structure d'une table

La clause ALTER permet la modification des colonnes d'un table. Associée avec la clause DROP, elle permet de supprimer des colonnes. La syntaxe est la suivante :

```
mysql>ALTER TABLE Nom_de_la_table DROP Nom_de_la_colonne
```

Il faut noter que la suppression de colonnes n'est possible que dans le cas où la colonne ne fait pas partie d'un index et n'est pas l'objet d'une contrainte d'intégrité.

Associée avec la clause ADD, la clause ALTER permet l'ajout de colonnes à une table. La syntaxe est la suivante :

```
mysql>ALTER TABLE Nom_de_la_table ADD (Nom_de_la_colonne Type_de_donnees)
```

Associée avec la clause MODIFY, la clause ALTER permet la modification du type de données d'une colonne. La syntaxe est la suivante :

```
mysql>ALTER TABLE Nom_de_la_table MODIFY (Nom_de_la_colonne Type_de_donnees)
```

Obtenir des informations sur les colonnes

La clause DESCRIBE renvoie la structure d'une table : liste des colonnes et leur type. La syntaxe est la suivante :

```
mysql> DESCRIBE nom_table
```

Exemple : Il s'agit de la structure de la table *user* contenue dans la base *mysql* :

```
mysql> describe user;
+-----------------+-----------------+------+-----+---------+-------+
| Field           | Type            | Null | Key | Default | Extra |
+-----------------+-----------------+------+-----+---------+-------+
| Host            | char(60) binary |      | PRI |         |       |
| User            | char(16) binary |      | PRI |         |       |
| Password        | char(16) binary |      |     |         |       |
| Select_priv     | enum('N','Y')   |      |     | N       |       |
| Insert_priv     | enum('N','Y')   |      |     | N       |       |
| Update_priv     | enum('N','Y')   |      |     | N       |       |
```

```
| Delete_priv     | enum('N','Y')   |      |     | N       |       |
| Create_priv     | enum('N','Y')   |      |     | N       |       |
| Drop_priv       | enum('N','Y')   |      |     | N       |       |
| Reload_priv     | enum('N','Y')   |      |     | N       |       |
| Shutdown_priv   | enum('N','Y')   |      |     | N       |       |
| Process_priv    | enum('N','Y')   |      |     | N       |       |
| File_priv       | enum('N','Y')   |      |     | N       |       |
| Grant_priv      | enum('N','Y')   |      |     | N       |       |
| References_priv | enum('N','Y')   |      |     | N       |       |
| Index_priv      | enum('N','Y')   |      |     | N       |       |
| Alter_priv      | enum('N','Y')   |      |     | N       |       |
+-----------------+-----------------+------+-----+---------+-------+
17 rows in set (0.00 sec)
```

3.3.2 Le langage de manipulation de données

SQL est un langage de manipulation de données (LMD), ela signifie qu'il permet de sélectionner, insérer, modifier ou supprimer des données dans une table d'une base de données relationnelle.

La sélection de données

La commande SELECT est basée sur l'algèbre relationnelle, n effectuant des opérations de sélection de données sur plusieurs tables relationnelles par projection. Sa syntaxe est la suivante :

```
mysql> SELECT <liste des noms de colonnes>
FROM <Liste des tables>
[WHERE <condition logique>]
```

- <liste des noms de colonnes> : liste des colonnes choisies, séparées par des virgules. Lorsque l'on désire sélectionner l'ensemble des colonnes d'une table il n'est pas nécessaire de saisir toute le liste des colonnes, l'option * permet de réaliser cette tâche.
- <Liste des tables> : liste des tables indique l'ensemble des tables (séparées par des virgules) sur lesquelles on opère.
- <condition logique> : permet d'exprimer des qualifications complexes à l'aide d'opérateurs logiques et de comparateurs arithmétiques.

Exemple :

La requête suivante permet d'afficher tout le contenu de la table ELEVE :

```
mysql> select * from ELEVE
```

NumEleve	NumClasse	Nom	Prenom	Adresse
1	1	Dupond	Marc	10 allée de Lilas
2	2	Dujardin	Muriel	42 rue des étangs
3	1	Bizé	Raphaël	1 rue du vignoble
4	2	Colomb	Jean-Marc	65 Grand Rue

Pour ne sélectionner que les élèves appartenant à la classe numéro 1, il faudra alors ajouter une condition :

```
mysql> select * from ELEVE where NumClasse=1
```

NumEleve	NumClasse	Nom	Prenom	Adresse
1	1	Dupond	Marc	10 allée de Lilas
3	1	Bizé	Raphaël	1 rue du vignoble

La sélection d'une partie des champs de la table (une projection) se fait de cette manière :

```
mysql> select Nom,Prenom from ELEVE where NumClasse=1
```

Nom	Prenom
Dupond	Marc
Bizé	Raphaël

Expression des restrictions

Une restriction consiste à sélectionner les lignes satisfaisant à une condition logique effectuée sur leurs attributs. En SQL, les restrictions s'expriment à l'aide de la clause WHERE suivie d'une condition logique exprimée à l'aide d'opérateurs logiques. Dans l'exemple précédent nous avons déjà vu le signe =, en voici d'autres :

- Les opérateurs logiques : AND, OR, NOT .
- Les comparateurs de chaîne : IN, BETWEEN, LIKE
- Les opérateurs arithmétiques : $+, -, *, /, \%$
- Les comparateurs arithmétiques : $:=, !=, >, <, >=, <=, <>, !>, !<$

Des jokers sont disponibles avec LIKE :

- Le caractère % permet de remplacer une séquence de caractères (éventuellement nulle)
- La caractère _ permet de remplacer un caractère
- Les caractères [-] permettent de définir un intervalle de caractères (par exemple [J-M])

Exemples : Sélection de tous les élèves dont le champ NumEleve est supérieur à 2 ET le champ NumClasse est égal à 1 :

```
mysql> select Nom,Prenom from ELEVE where NumClasse=1 and NumEleve>2
```

Sélection de tous les élèves dont le nom commence par "Du" :

```
mysql> select Nom,Prenom from ELEVE where Nom like "Du%"
```

Tri du résultat

Il existe d'autres options pour la commande SELECT qui permettent de trier le résultat d'un requête :

GROUP BY : regrouper le résultat de la sélection en fonction de la colonne passée en paramètre.

ORDER BY : classe dans l'ordre de la colonne citée.

Exemples : La commande ci-dessous extrait tous les élèves de la table, les regroupe par classe, et les classe dans l'ordre alphabétique des noms :

```
mysql> select * from ELEVE group by NumClasse order by NomEleve
```

NumEleve	NumClasse	Nom	Prenom	Adresse
3	1	Bizé	Raphaël	1 rue du vignoble
1	1	Dupond	Marc	10 allée de Lilas
4	2	Colomb	Jean-Marc	65 Grand Rue
2	2	Dujardin	Muriel	42 rue des étangs

Expression de jointures

En SQL, l'expression d'une jointure se fait en précisant le nom des colonnes des tables sur lesquelles on fait la jointure. On désigne les colonnes des différentes tables en écrivant le nom de la table, suivie d'un point puis du nom de la colonne. La clause WHERE permet de préciser la qualification (condition) de la jointure.

Exemple :

La table NOTE contient les données ci-dessous :

NumEleve	NumMatiere	Note
1	1	12
1	5	16
1	7	10
2	1	13
2	5	14

La sélection de toutes les notes des élèves se fait grâce à la requête ci-dessous :

```
mysql> select ELEVE.Nom, ELEVE.Prenom, NOTE.Note from ELEVE,NOTE
where ELEVE.NumEleve=NOTE.NumEleve
```

On obtiendra le résultat suivant :

Nom	Prenom	Note
Dupond	Marc	12
Dupond	Marc	16
Dupond	Marc	10
Dujardin	Muriel	13
2	5	14

Comment saisir une chaîne de caractères avec MySQL

Une chaîne de caractères est une liste de caractères encadrés par des apostrophes (' ') ou des guillemets (" "). A l'intérieur de cette chaîne de caractères, certains d'entre eux peuvent avoir une signification particulière pour le serveur, il faut alors les précéder d'un \.

- Le caractère ' devra être remplacé par \ '
- Le caractère " devra être remplacé par \ "
- Un antislash sera inséré de cette manière : \ \
- Un % sera saisi ainsi \ %
- L'insertion de ce caractère \ n correspond à un retour à la ligne
- Le caractère \ r sera interprété comme un retour chariot

Insertion de données

L'insertion de nouvelles données dans une table se fait grâce à l'ordre INSERT, qui permet d'insérer de nouvelles lignes dans la table.

```
mysql>INSERT INTO nom_table [(nom_col,...)] VALUES (expression,...)
```

La liste des colonnes à remplir est optionnelle. Si elles ne sont pas précisées, MySQL considèrera qu'il doit remplir toutes les colonnes, dans ce cas là il doit y avoir autant de données passées en paramètres dans la section VALUES que de colonnes dans la table, sinon une erreur sera renvoyée par le serveur et l'insertion n'aura pas lieu.

Exemple : Ajout de la note 17 à Muriel Dujardin (dont le NumEleve=2), dans la matière numéro 7 :

```
mysql>INSERT INTO NOTE VALUES (2,5,17)
```

Modification de données

La modification de données (aussi appelée mise à jour) consiste à modifier des lignes dans une table grâce à l'ordre UPDATE. La modification à effectuer est précisé après la clause SET. Il s'agit d'une affectation d'une valeur à une colonne grâce à l'opérateur = suivi d'une expression algébrique, ou d'une constante. La clause WHERE permet de préciser les enregistrements sur lesquels la mise à jour aura lieu. La syntaxe de cette clause est :

```
mysql>UPDATE Nom_de_la_table
SET Colonne = Valeur_Ou_Expression
WHERE <condition logique>
```

Exemple : On veut modifier la note de Muriel Dujardin dans la matière numéro 7, elle doit valoir 18 :

```
mysql>UPDATE NOTE
SET Note = 18
WHERE NumEleve=2 and NumMatiere=7
```

Suppression de données

La suppression de données dans une table se fait grâce à l'ordre DELETE. Celui-ci est suivi de la clause FROM, précisant la table sur laquelle la suppression s'effectue, puis d'une clause WHERE qui décrit la qualification, c'est-à-dire l'ensemble des lignes qui seront supprimées.

L'ordre DELETE est à utiliser avec précaution car l'opération de suppression est irréversible. Il faudra donc s'assurer dans un premier temps que les lignes sélectionnées sont bien les lignes que l'on désire supprimer !

La syntaxe est la suivante :

```
mysql>DELETE FROM table_name [WHERE <condition logique>]
```

ATTENTION ! Si la condition n'est pas définie dans la section where, tous les enregistrements de la table seront effacés.

Atelier 3 : Conception et création d'une base de données

Exercice 3.1 *Conception d'une base de données*

Nous voulons gérer les commandes passées par nos clients.

Chacun de nos clients est identifié par son nom, son adresse, sa ville, l'état de se compte et son code client. Les produits que nous vendons sont référencés par leur numéro de produit, leur libellé, leur prix et la quantité qu'il en reste en stock. Chaque ligne de nos bons de commandes contient la référence et le libellé du produit, la quantité désirée, le prix unitaire, et le prix total (PU * quantité). La date de la commande doit être aussi indiquée au bas de la commande.

1. Listez les données.
2. Etablissez le Modèle Conceptuel de données
3. Ecrivez le modèle logique de données

Exercice 3.2 *Découverte du langage SQL*

1. Créez dans la base **test** une table nommée **eleve**, contenant les champs :
   - cle, un entier non signé qui s'autoincrémente
   - prenom, de type chaine de caractères
   - nom, de type chaine de caractères
   - adresse, chaine de caractères
   - date_de_naissance, de type date
   - classe, de type chaine de caractères

   Vous devez la voir apparaitre dans la liste renvoyée par la commande **show tables ;** et trois fichiers ont été créés dans le répertoire de la base de données **test**, à savoir *DATA-DIR/var/test*.
2. . Remplissez cette table avec le contenu suivant :
   - jean ; dupont ; 15 av des lilas - 54000 NANCY ;15/12/1996 ; CP ;
   - marc ; grosjean ; 3 rue du passé -75000 PARIS ;24/03/1990 ; CE1 ;
   - julie ; durand ; 69 grand rue - 88370 PLOMBIERES LES BAINS ; 31/12/1995 ;CE1 ;
3. . Recherchez tous les élèves ayant plus de 7 ans.
4. . La date de naissance de Marc Grosjean n'est pas la bonne, mettez-la à jour avec celle-ci : 24/03/1995.
5. . Modifiez la structure de la table en ajoutant le champ **telephone** juste après le champ adresse.

4

Administration de MySQL

Objectifs

- Savoir configurer le serveur MySQL.
- Etre capable de gérer les accès aux différentes bases de données.
- Sauvegarder et restaurer tout ou partie d'une base de données.

Contenu

4.1 Présentation du fichier my.cnf 46
4.2 Configuration de MySQL 47
4.3 La gestion des privilèges 52
4.4 Sauvegardes et restitutions 57
Atelier 4 61

Références

- Le manuel de référence de MySQL situé à l'adresse : http://www.mysql.com/documentation/mysql/bychapter/

Dans ce chapitre nous allons aborder la configuration du serveur MySQL, du client texte, et l'utilisation de quelques outils de maintenance du serveur. La configuration du serveur est relativement simple : tous les paramètrages se font en modifiant un seul et même fichier : **my.cnf**

4.1 Présentation du fichier my.cnf

Les concepts du fichier de configuration de MySQL.

Le fichier de configuration de MySQL est un fichier qui porte le nom de `my.cnf`. On peut le placer à plusieurs endroits sur le système de fichiers, en fonction de ce qu'on veut configurer :

- */etc/my.cnf :* pour définir une configuration globale,
- */DATA-DIR/my.cnf :* pour définir les options du serveur,
- */.my.cnf :* pour définir les options spécifiques à un utilisateur.

Ce fichier est divisé en sections représentées par des mots entre crochets. Le nom d'une section correspond au nom de la commande, ou programme concerné. Chaque section se termine là où commence la suivante. Différents types d'options peuvent être définis :

```
[section]
option0
option1      = valeur1
set-variable = nom_variable = valeur2
```

Les deux premières lignes permettent d'ajouter des options :

- **option0** est équivalent à **--option0** en ligne de commande.
- **option1=valeur1** est équivalent à **--option1=valeur1** en ligne de commande.

La troisième permet de définir une variable :

- **set-variable=nom_variable=valeur2** est équivalent à **--set-variable nom_variable=valeur2** en ligne de commande.

Exemple, la section [mysqld] comprend les options du serveur et la section [client] celles du client mysql.

Quatre modèles de ce fichier sont disponibles dans le répertoire *MySQL-DIR/share/mysql* de la version compilée (ou dans le répertoire /usr/share/mysql pour la version en paquetages). Ils sont nommés *my-small.cnf, my-medium.cnf, my-large.cnf et my-huge.cnf*, utiliser celui qui correspond le mieux à la charge du serveur.

4.2 Configuration de MySQL

Configuration du serveur, du client et des outils MySQL à l'aide du fichier my.cnf.

La configuration du serveur MySQL se fait en passant différents paramètres au démon **mysqld**.

4.2.1 Les options de mysqld

Dans le fichier de configuration **my.cnf**, la section [**mysqld**] comprend les paramètres du serveur sous forme d'options ou de variables.

Les options orientées système

-b *répertoire* **ou --basedir=***répertoire*
répertoire est le répertoire d'installation de MySQL.

--bind-address=IP
adresse IP à associer au serveur.

-h *répertoire* **ou --datadir=***répertoire*
répertoire est le répertoire où se trouvent les bases de données.

--pid-file=*chemin*
chemin vers le fichier pid.

-P *numero_port* **ou --port=***numero_port*
numéro de port que le serveur va écouter.

--one-thread
le serveur n'utilisera qu'un seul thread. Utile lors du débogage sous Linux.

--socket=*chemin*
chemin du fichier de socket utilisé pour se connecter en local.

-t *répertoire* **ou --tmp-dir=***répertoire*
définition du répertoire temporaire.

-u *nom_utilisateur* **ou --user=***nom_utilisateur*
nom de l'utilisateur Linux qui va exécuter mysqld, pour des raisons de sécurité, il est conseillé de lancer mysqld en qu'utilisateur **mysql**, qui est celui par défaut.

Les paramètres locaux

--character-sets-dir=*répertoire*
répertoire est le répertoire où se trouvent les jeux de caractères.

--default-character-set=*jeu_caracteres*
définition du jeu de caractères à utiliser.

-L *langue* **ou --language**=*langue*
langue à utiliser pour les messages d'erreur envoyés aux clients.

Les options de supports

--ansi
utilise la syntaxe ANSI SQL au lieu de la syntaxe MySQL.

--default-table-type=*type*
type de tables à utiliser par défaut. Plus d'informations sur les différents types de tables au chapitre 5.1.

Les options de sécurité

--safe-show-database
ne montre pas à un utilisateur connecté les bases de données pour lesquelles il n'a aucun privilège.

--skip-networking
seules les connexions via socket Unix sont autorisées. Recommandé pour des serveurs n'ayant que des requêtes depuis la machine locale.

--skip-show-database
n'autorise la requête `show databases` que si l'utilisateur a le privilège **process**.

Les fichiers de log

-l ou --log=*fichier*
fichier est le fichier de log des connections et requêtes.

--log-slow-queries=*fichier* loggue les requêtes dont la durée est supérieure à **long_query_time** dans le fichier passé en paramètre.

Les variables

Les variables sont définies grâce à l'option `--set-variable` :

key_buffer_size : les blocs d'index sont stockés dans des buffers et partagés par tous les threads. Cette variable permet de définir la taille de ce buffer.

long_query_time : utile si on active `--log-slow-queries`, cette variable définit la durée minimum des requêtes à tracer.

max_allowed_packet : taille maximum d'un paquet transitant entre le serveur et les clients. Avec MySQL-3.23.xx, la taille maximum d'un paquet est de 16Mo, avec MySQL-4.0, elle est de 4Go.

max_connections : nombre maximum de connexions simultanées. Par défaut cette variable est égale à 100.

sort_buffer : buffer utilisé pour les requêtes comportant un "group_by" ou "order_by".

table_cache : nombre de tables ouvertes pour tous les threads.

wait_timeout : nombre de secondes que le serveur attend avant de fermer une connexion.

Exemple

Ci-dessous un exemple de section [**mysqld**] du fichier *my.cnf* :

```
# The MySQL server
[mysqld]
port            = 3306
socket          = /tmp/mysql.sock
set-variable    = key_buffer=16M
set-variable    = max_allowed_packet=1M
set-variable    = table_cache=64
set-variable    = sort_buffer=512K
set-variable    = net_buffer_length=8K
set-variable    = myisam_sort_buffer_size=8M
language        = french
```

Dans cet exemple, on voit, entre autres, que les messages d'erreur envoyés au client sont envoyés en français, que le fichier de socket se trouve dans le répertoire */tmp/-mysql.sock* et que le port de connexion en TCP/IP est 3306.

4.2.2 Les options de mysql

De même que pour le serveur, il est possible de configurer le client mysql en passant les options directement sur la ligne de commande, ou grâce au fichier my.cnf. Toutes les options de configuration du client sont disponibles en exécutant la commande :

```
shell>mysql --help
```

Parmi ces options on trouvera :

Les options concernant l'utilisateur

-u *nom_user* **ou --user=***nom_user*
nom d'utilisateur mysql.

-p *mot_de_passe* **ou --password=***mot_de_passe*
mot de passe de l'utilisateur.

Les options relatives au serveur

-h *adresse_serveur* **ou --host=***adresse_serveur*
adresse IP ou nom d'hôte du serveur mysql.

-P ou --port=*numero_port*
port de connexion TCP/IP du serveur mysql

-S ou --socket=*chemin_fichier_socket*
fichier de socket à utiliser pour se connecter au serveur MySQL

-D ou --database=*database_name*
Nom de la base de données à laquelle se connecter.

Les paramètres de connexion

-C ou --compress
Utilise un protocole compressé entre le client et le serveur : économise de la bande passante

-H ou --html
Renvoie les résultats au format HTML.

-t ou --table
Renvoie les résultats sous forme de tableau.

--character-sets-dir=*répertoire_jeu_caracteres*
Répertoire où se trouvent les jeux de caractères.

La liste ci-dessous correspond aux variables que l'on peut définir de manière à configurer la connexion entre le client et le serveur.

* **connect_timeout :** nombre de secondes avant la fermeture de la connexion.
* **max_allowed_packet :** taille maximale d'un paquet transmis entre le client et le serveur. (16 Mo par défaut) .
* **net_buffer_size :** taille du buffer pour une connexion TCP/IP ou socket. (16 Ko par défaut).

Exemple

La section [**client**] du fichier my.cnf :

```
[client]
user = dupont
host = ADRESSE_SERVEUR
port = 3306
```

et son équivalent en ligne de commande avec, en plus, l'option -p de manière à se connecter avec mot de passe :

```
shell>mysql -u dupont -h ADRESSE_SERVEUR -P 3306 -p
Enter password :
mysql>
```

4.2.3 Administration du serveur avec mysqladmin

L'administration de MySQL se fait avec l'utilitaire `mysqladmin`. Sa syntaxe est la suivante :

```
mysqladmin [options] commande paramètres
```

- Les options sont similaires aux options du client shell.
- Les principales commandes de `mysqladmin` sont :

  create nom_base : créer une base de données de nom 'nom_base'.

  drop nom_base : supprimer une base de données de nom 'nom_base'.

  password le_nouveau_mot_de_passe : changer le mot de passe.

  ping : teste si mysqld fonctionne.

  flush-logs, flush-hosts, flush-tables, flush-privileges : rafraîchit les logs, les hôtes, les tables ou tout.

  shutdown : arrête le serveur de base de données.

  status : donne un état du serveur de base de données.

  version : donne la version du serveur.

Exemples : La commande ci-dessous permet de définir un mot de passe à l'utilisateur `root` :

```
shell>mysqladmin -u root password NOUVEAU_MOT_DE_PASSE
```

et celle-ci permet de créer une nouvelle base de données, appelée GESTION :

```
shell>mysqladmin -u root -p create GESTION
```

4.3 La gestion des privilèges

Comprendre le fonctionnement de l'authentification sous MySQL et savoir gérer les accès aux différentes bases de données.

4.3.1 Introduction

Il est vital, lors de la mise en place d'un système d'informations, de bien gérer les accès aux différentes bases de données. En effet les données étant au centre du fonctionnement d'une entreprise, seules les personnes étant amenées à manipuler, modifier, supprimer des données doivent être autorisées à le faire.

Les privilèges, sous MySQL, sont les différents droits que l'on peut ou non accorder aux différents utilisateurs.

4.3.2 Fonctionnement

Lors d'une connexion à un serveur MySQL, le contrôle d'accès se fait en deux étapes : la première au moment de l'établissement de la connexion, puis ensuite lors de l'exécution de chaque requête.

Identification à la connexion

L'identification est basée sur deux informations : le nom d'utilisateur et l'adresse depuis laquelle cet utilisateur se connecte.

Les informations concernant les utilisateurs et les privilèges qui leur sont accordés sont stockés dans une base de données, nommée **mysql**. Cette base comprend notamment la table **user**, dans laquelle sont stockés tous les utilisateurs ayant le droit de se connecter au serveur.

Lorsqu'un client fait une demande de connexion au serveur, celui compare les nom d'utilisateur, mot de passe et nom de domaine, que le client lui a envoyés, avec les informations qui se trouvent dans la table **user**

La table **user** est triée en fonction du nom de domaine, du plus précis au moins précis. Si le même nom de domaine est défini plusieurs fois, la table est retriée en fonction du nom d'utilisateur, du plus précis au moins précis.

C'est à partir de cette table triée que va être effectuée l'identification. La première entrée qui correspond aux paramètres fournis par le client sera celle utilisée pour identifier l'utilisateur. Si en revanche aucune entrée ne correspond, la connexion sera refusée.

Exemple : Supposons que notre table *user* contienne les données suivantes :

```
+-----------+----------+-
| Host      | User     | ...
+-----------+----------+-
| %         | root     | ...
| %         | jeffrey  | ...
| localhost | root     | ...
| localhost |          | ...
+-----------+----------+-
```

C'est à partir de la table triée ci-dessous que le serveur va procéder à l'authentification :

```
+-----------+----------+-
| Host      | User     | ...
+-----------+----------+-
| localhost | root     | ...
| localhost |          | ...
| %         | jeffrey  | ...
| %         | root     | ...
+-----------+----------+-
```

Si jeffrey tente de se connecter depuis localhost, le serveur va d'abord regarder parmi les lignes dont le nom de domaine est localhost, à savoir :

```
+-----------+----------+-
| Host      | User     | ...
+-----------+----------+-
| localhost | root     | ...
| localhost |          | ...
+-----------+----------+-
```

Parmi ces deux lignes, c'est celle dont la colonne *User* est vide qui va correspondre et permettre à jeffrey de se connecter au serveur. La ligne contenant jeffrey comme nom d'utilisateur, et % comme nom de domaine aurait fonctionné aussi, mais ce n'est pas la première ligne à correspondre dans la table.

Contrôle d'accès

Une fois la connexion établie, le serveur s'assure pour chaque requête à effectuer que l'utilisateur a bien accès à la base de données, les tables et les colonnes concernées, et surtout qu'il a bien les droits nécessaires sur celles-ci pour pouvoir effectuer la requête.

De même que pour les informations d'authentification, les privilèges accordés aux utilisateurs sont stockés dans les tables de la base de données **mysql**. Elles sont au nombre de cinq et agissent à différents niveaux :

au niveau global : les privilèges attribués de manière globale à un utilisateur sont valables pour toutes les bases de données. On préférera garder ce niveau de privilège pour les super utilisateurs.

au niveau base de données : les droits accordés à ce niveau ne s'appliquent que pour la base de données concernée.

au niveau des tables et des colonnes : pour chaque table et chaque colonne, il est possible de définir quels privilèges sont attribués aux utilisateurs.

De la même manière que pour la phase d'authentification, les tables dans lesquelles sont stockées les informations de privilèges, sont triées en fonction du nom de domaine et du nom d'utilisateur ; et c'est à partir de ces tables triées que le serveur vérifiera si l'utilisateur a bien les droits suffisants pour exécuter la requête.

Il est possible de gérer les droits en manipulant directement ces tables, mais cette pratique est déconseillée. On utilisera plutôt les commandes **GRANT** et **REVOKE** pour ajuster les droits.

4.3.3 Ajout de privilèges

GRANT permet de donner des droits à un utilisateur. Si cet utilisateur n'existe pas, il sera créé, c'est donc également la commande qui permet de créer des utilisateurs. Les utilisateurs et les droits créés par la commande GRANT sont enregistrés dans les différentes tables de la base **mysql**.

```
mysql> GRANT commande ON nom_base.nom_table TO user@hostname [IDENTIFIED BY
'password'];
```

commande	Cette valeur peut être ALL PRIVILEGES, FILE, RELOAD, ALTER, INDEX, SELECT, CREATE, INSERT, SHUTDOWN, DELETE, PROCESS, UPDATE, DROP, REFERENCES, USAGE . Si plusieurs commandes doivent être présentes dans cette ligne de commande, les séparer par des virgules.
nom_base	Nom de la base à laquelle a accès l'utilisateur ou '*' pour toutes.
nom_table	Nom de la table à laquelle a accès l'utilisateur ou '*' pour toutes.
user	Nom de l'utilisateur (16 caractères maximum).
hostname	Nom de l'hôte ou '%' pour tous. Attention, MySQL identifie un utilisateur par son nom **et** son hôte.
password	Le mot de passe de l'utilisateur.

Exemple : création d'un utilisateur dupont, qui n'a que le droit "select" sur la table "intranet.news", et qui peut se connecter depuis toutes les machines dont l'adresse commence par "192.168.0."

```
mysql> GRANT
SELECT ON intranet.news TO dupont@"192.168.0.%" IDENTIFIED BY 'azerty';
```

4.3.4 Suppression de privilèges

REVOKE réalise l'opération inverse de GRANT : il retire des droits. Les paramètres sont les mêmes que ceux de GRANT.

```
mysql> REVOKE commande ON nom_base.nom_table FROM user@hostname;
```

4.3.5 Mots de passe

Pour définir un mot de passe, on dispose de la commande MySQL **set password** et de la commande shell **mysqladmin** (Voir paragraphe 4.2.3) :

```
shell> mysql -u root mysql
mysql> SET PASSWORD FOR user_name@hostname=PASSWORD('new_password');
```

```
shell> mysqladmin -u user_name -p password new_password
Enter password:
```

4.3.6 Vérification des privilèges attribués

La commande shell **mysqlaccess** permet de lister les privilèges attribués à un domaine, un utilisateur de ce domaine sur une base données.

La syntaxe

```
shell>mysqlaccess [domaine [nom_utilisateur [nom_base]]] OPTIONS
```

Parmi les options de cette commande, on retrouve les suivantes :

-u : nom d'utilisateur permettant de se connecter au serveur.

-p : mot de passe.

-h : nom de domaine ou adresse IP du serveur.

-U : se connecter en tant que superutilisateur.

-P : mot de passe du superutilisateur.

-b : résume tous les privilèges dans un seul tableau.

-d : nom de la base de données dont on veut vérifier les accès.

–help : affiche l'aide de mysqlaccess.

Exemple

```
shell> mysqlaccess -U root -P -d mysql -b
Password for MySQL superuser root:

Sele Inse Upda ... Shut Proc File Gran Refe Inde Alte | Host,User,DB
---- ---- ---- ... ---- ---- ---- ---- ---- ---- ---- + --------------------
 N    N    N   ...  N    N    N    N    N    N    N   | localhost,toto,mysql
 N    N    N   ...  N    N    N    N    N    N    N   | localhost,moi,mysql
 Y    Y    Y   ...  Y    Y    Y    Y    Y    Y    Y   | localhost,root,mysql
 N    N    N   ...  N    N    N    N    N    N    N   | localhost,ANY_NEW_USER,my
```

Sur ce système, aucun utilisateur, hormis `root`, n'a de privilège sur la base `mysql`.

4.4 Sauvegardes et restitutions

Savoir sauvegarder tout ou partie d'une base de données. Etre capable de restaurer ces données.

Comme pour tout système d'information, la sauvegarde des bases de données MySQL peut s'avérer vitale. Différentes méthodes de sauvegardes sont disponibles.

4.4.1 Sauvegarde avec mysqldump

Cette commande permet de sauvegarder tout ou partie d'une base de données, voire toutes les bases de données d'un serveur MySQL. La sauvegarde générée par cette commande est en fait la liste des requêtes à exécuter pour pouvoir reconstruire la base de données ou la table sauvegardées.

La syntaxe

```
shell> mysqldump [OPTIONS] BDD [tables]
ou
shell> mysqldump [OPTIONS] --databases [OPTIONS] BDD1 [BDD2 BDD3...]
ou
shell> mysqldump [OPTIONS] --all-databases [OPTIONS]
```

Les options le plus couramment utilisées :

--add-locks : verrouille les tables avant et déverrouille après la sauvegarde de chaque table.

--all-databases : sauvegarde toutes les bases de données.

--databases : suivie de la liste des bases de données à sauvegarder.

-r fichier : fichier dans lequel sera stocké la sauvegarde.

--tables : Sauvegarde les tables de la base passée en paramètre

-T : cette option sauvegarde chaque table dans deux types de fichier :

- nom_table.txt : contient les données de la tables séparées suivant les options `--field-XXX` et `--lines-XXX`
- nom_table.sql : contient la requête nécessaire à la recréation de la table.

--fields-terminated-by=... ; --fields-enclosed-by=... ; --fields-optionally-enclosed-by=... ; --fields-escaped-by=... ; --lines-terminated-by=... Ces champs permettent de définir les séparateurs dans les fichiers de données lorsqu'on ne sauvegarde que certaines tables avec l'option `-T` .

--opt : équivalent à mettre les options : --quick --add-drop-table --add-locks --extended-insert --lock-tables. Permet de faire une sauvegarde plus rapide.

--quick : ne sauvegarde pas en passant par un buffer, écrit directement le résultat sur stdout.

--add-drop-table : ajoute la commande `drop table` avant chaque requête de création de table.

--extended-insert : utilise la commande INSERT en mode multiligne : plus compact et plus rapide.

--lock-tables : verrouille toutes les tables en READ LOCAL avant de commencer la sauvegarde. Ceci autorise des insertions concurrentes. A noter que dans le cas de sauvegarde de plusieurs bases de données, les verrous sur les tables sont posés de manière totalement indépendante, ce qui peut poser des problèmes d'intégrité si les tables de bases différentes ne sont pas sauvegarder dans le même état.

Options de connexion : les options de connexion sont les mêmes que celles du client mysql : **-u, -p, -h, -P, -S**.

Exemple

Nous avons une base de données GESTION, à l'intérieur de laquelle quatre tables : CLIENT, COMMANDE, FOURNISSEUR, LIGNE_COMMANDE.

Une première méthode de sauvegarde consiste à tout sauvegarder dans le même fichier GESTION.sql :

```
shell>mysqldump -u root -p GESTION > /home/sauvegarde/GESTION.sql
```

On peut également sauvegarder chaque table de manière indépendante :

```
shell>mysqldump -u root -p GESTION -T /home/sauvegarde/
```

On aura alors 8 fichiers dans le répertoire */home/sauvegarde* : CLIENT.txt, CLIENT.sql, COMMANDE.txt, COMMANDE.sql, FOURNISSEUR.txt, FOURNISSEUR.sql, LIGNE_COMMANDE.txt, et LIGNE_COMMANDE.sql.

4.4.2 Sauvegarde avec "Select... into outfile ..."

Il est possible de rediriger le résultat d'une requête SQL **SELECT** dans un fichier.

La syntaxe

```
mysql>select * from nom_table into outfile '/chemin/vers/fichier.txt'
```

S'assurer que le serveur a bien le droit d'écriture à l'endroit où enregistrer le fichier destination.

Cette méthode permet de sauvegarder les données d'une table de manière indépendante, mais égalemant le résultat de recherche d'informations provenant de plusieurs tables.

Exemple

La sauvegarde de la table COMMANDE de la base GESTION dans un fichier situé dans */tmp* s'effectue de cette manière :

```
mysql>select * from COMMANDE into outfile '/tmp/COMMANDE.txt'
```

4.4.3 Sauvegarde avec la commande tar

Avec MySQL, il est possible d'effectuer une sauvegarde en utilisant la commande tar. Il suffit d'archiver les dossiers correspondant aux bases de données à sauvegarder :

```
shell>tar cvf nom_base.tar nom_base
```

La restauration s'effectue alors simplement en désarchivant le fichier .tar dans le répertoire des données.

4.4.4 Restauration

Dans le cas d'une sauvegarde en un seul fichier contenant les requêtes de création et remplissage des tables, la restauration se fait de cette manière :

```
shell>mysql -u root -p GESTION < GESTION.sql
```

Pour pouvoir restaurer une base entière, exécuter mysql avec le fichier de sauvegarde en entrée. Si la sauvegarde a été réalisée avec l'option `--databases`, il n'y a pas besoin de créer la base avant de pouvoir restaurer les tables qui la composent, et il n'y a pas non plus besoin de préciser au client mysql quelle base utiliser (la commande USE NOM_BASE précède les commandes de chaque base de données se trouvant dans le fichier).

Dans le cas d'une sauvegarde avec l'option **-T**, il faut d'abord exécuter le fichier .sql, avec la commande **source** du client mysql ou comme ci-dessus en passant le fichier .sql en paramètre à mysql et en précisant la base de données à utiliser. Puis il faut importer les données contenues dans le fichier .txt avec la commande suivante :

```
mysql> LOAD DATA INFILE nom_fichier.sql INTO TABLE nom_table;
```

Il existe également la commande shell **mysqlimport**, qui est l'équivalent de LOAD DATA INFILE. Cette commande s'utilise de la manière suivante :

```
shell> mysqlimport [OPTIONS] nom_base fichier1.txt [fichier2.txt ...];
```

Pour chaque fichier, MySQL importe le contenu du fichier dans la table ayant le nom de celui-ci sans son extension. Exemple, le fichier GESTION.txt sera importé dans la table GESTION.

Quant aux options, la plupart sont les mêmes que celles de mysqldump. Elles sont disponibles en exécutant `mysqlimport --help`.

Atelier 4 : Administration de MySQL

Exercice 4.1 *Gestion de privilèges*

1. Mettez un mot de passe à l'utilisateur root de MySQL.
2. Ajoutez un utilisateur **paul**, qui a le droit de se connecter uniquement depuis la machine locale et qui n'a que le droit SELECT sur la table test.eleve.
3. Créez un utilisateur **pierre** qui peut se connecter depuis toutes les machines et qui n'a que le droit SELECT sur toute la base test.
4. Donnez à paul le droit de se connecter depuis une autre machine, et accordez-lui les droits INSERT, UPDATE et DELETE. Vérifiez.
5. Supprimer l'utilisateur pierre. Vérifiez qu'il ne peut plus se connecter.

Exercice 4.2 *Sauvegarde et restauration*

1. A l'aide de la commande **mysqldump**, sauvegardez la base de données test.
2. Utilisez la commande **mysqladmin** pour créer la base de données **test2**
3. Réimportez la dans la base **test2**.

Exercice 4.3 *MySQL utilisable uniquement en local*

En ajoutant l'option skip-networking dans le fichier my.cnf, vérifiez que vous ne pouvez plus vous connecter que depuis la machine locale.

5

Optimisation et administration avancée de MySQL

Objectifs

- Connaître les différents types de tables supportés par MySQL.
- Savoir mettre en place un système de réplication.
- Comprendre le fonctionnement et le rôle des index.

Contenu

5.1 Les types de tables 63
5.2 La réplication 74
5.3 De la bonne utilisation des index 81
Atelier 5 83

Références

- Le manuel de référence de MySQL situé à l'adresse : http://www.mysql.com/documentation/mysql/bychapter/
- http://www.innodb.com, le site de référence des tables de type InnoDB.

5.1 Les types de tables

Quels types de tables sont disponibles avec MySQL ? Lequel choisir ?

Depuis la version 3.23.6, il est possible de choisir entre plusieurs types de tables. D'un côté on a les types de tables ne supportant pas le mode transactionnel : ISAM, MyISAM, HEAP, MERGE ; et de l'autre on a les types de tables apportant les transactions sécurisées : InnoDB et Berkeley DB.

Avantages des tables ne supportant pas les transactions :

- Elles sont plus rapides car elles n'ont pas à gérer de système de transaction.
- Elles utilisent moins d'espace disque puisqu'elles n'ont justement pas à gérer de transaction.
- Elles ont besoin de moins de ressources mémoire lors d'une mise à jour.

Avantages des tables transactionnelles :

- Il est possible de réaliser plusieurs requêtes et de les valider toutes en même temps grâce à un `COMMIT`.
- Il est possible d'annuler toutes les requêtes effectuées dans la transaction à l'aide d'un `ROLLBACK`.
- Si une mise à jour échoue au milieu de son exécution, tous les changements effectués seront annulés.

Chaque type de table ayant ses propres caractéristiques, il peut être intéressant de les combiner au sein d'une même base.

- Les tables MyISAM pour contenir des données "listées",
- Les tables HEAP pour les tables temporaires,
- Les tables MERGE pour stocker des logs,
- Les tables InnoDB pour avoir les supports de mode transactionnel et de clés étrangères.

Quelque soit le type de table choisi, lors de la création d'une table un fichier *nom_table.frm* est créé dans le répertoire de la base de données pour contenir sa structure.

5.1.1 ISAM

Le type **ISAM** est devenu obsolète. Il va bientôt disparaître parce que le type MyISAM en est une meilleure implémentation.

Les index sont stockés dans des fichiers .ISM et sont de type B-Tree.
Les données sont stockées dans des fichiers .ISD.

Les plus grosses différences entre les tables ISAM et MyISAM sont les suivantes :

- Les tables ISAM ne sont pas portables vers des plates-formes ayant un autre système d'exploitation.
- Les tables ISAM ne peuvent pas être supérieures à 4 Go.
- Les tables dynamiques sont plus fragmentées avec ISAM.
- La compression d'une table ISAM se fait avec **pack_isam**.

Pour changer le type d'une table ISAM vers MyISAM, de manière à pouvoir se servir d'outils tels que *mysqlchk*, utiliser la commande ALTER :

```
mysql> ALTER TABLE tbl_name TYPE = MYISAM;
```

5.1.2 MyISAM

Présentation

C'est le type de table par défaut pour MySQL-3.23. Il est basé sur le type ISAM, et possède de nombreuses extensions.

Les index sont de type B-Tree et sont stockés dans un fichier dont l'extension est .MYI (pour MyIndex).
Les données sont stockées dans des fichiers d'extension .MYD (pour MyData)

Les tables MyISAM ont différents formats dépendant du type de données qu'elles contiennent. Parmi eux les types STATIC et DYNAMIC expliqués ci-après.

Le format de table STATIC

Le format **STATIC** est le format par défaut. Il est utilisé quand la table ne contient aucun champ de taille variable BLOB, TEXT ou VARCHAR.

C'est le format le plus simple et le plus sécurisé. En effet, il est facile de retrouver des données, lors d'une recherche avec un index, quand la longueur d'une ligne est constante, puisqu'il suffit de multiplier la longueur d'une ligne par le nombre de lignes pour savoir où elles se situent dans le fichier.

Dans le même esprit, la réparation d'une table avec myisamchk est elle aussi plus sûre, puisque il est facile dans ce cas de savoir où commence et où se termine une ligne.

Le format de table DYNAMIC

Le format **DYNAMIC** est utilisé si la table contient des champs de type *VARCHAR, BLOB ou TEXT*, ou si la table a été créée avec l'option *ROW_FORMAT=DYNAMIC*.

Ce format est un peu plus compliqué dans la mesure où chaque ligne doit comporter une entête précisant sa taille. Une ligne peut également être stockée en plusieurs endroits dans le cas où, par exemple, lors d'une commmande UPDATE l'espace restant n'était pas suffisant.

Il est possible de défragmenter ce type de table avec l'outil **myisamchk**, ou la commande **OPTIMIZE TABLE**.

Il n'est pas très facile de reconstruire une table de ce format après un plantage, parce que les lignes sont fragmentées, chaque fragment est aussi appelé un lien , et il se peut que des liens manquent.

Forcer le type d'une table

Lors de la création d'une table de type MyISAM n'ayant pas de champ de type BLOB, on peut forcer le format à **DYNAMIC** ou **FIXED** avec l'option **ROW_FORMAT=** de la commande **CREATE TABLE**.

Options à passer à mysqld

Certaines options peuvent être passées au serveur soit en ligne de commande, soit à l'aide du fichier my.cnf détaillé au chapitre 4.2.

--myisam-recover[=option [,option]...]
Réparation automatique d'une table après un plantage ou si elle est mal fermée. Les options sont à choisir parmi :
- DEFAULT : revient à ne pas mettre d'options.
- BACKUP : si la table a été modifiée pendant la réparation, effectue une sauvegarde portant le nom nom_table-dateheure.BAK .
- FORCE : effectue la restauration même si il y a un risque de perdre plus d'une ligne de données.

-O myisam_sort_buffer_size=*taille_buffer*
Buffer utilisé pendant la réparation d'une table.

Maintenance des tables avec les commandes mysql

Même si les tables MyISAM ont été énormément testées, il reste néanmoins quelques circonstances pouvant les corrompre :

- Le démon `mysqld` a été tué pendant le traitement d'une commande d'écriture.
- L'arrêt inopiné du serveur.
- Utilisation d'outil comme myisamchk sur une table en cours d'utilisation.

Des commandes mysql existent pour vérifier l'état des tables : **CHECK TABLE**, et pour les réparer **REPAIR TABLE**.

Exemple :

```
 mysql> check table nom_table;
+----------+-------+----------+----------+
| Table    | Op    | Msg_type | Msg_text |
+----------+-------+----------+----------+
|nom_table | check | status   | OK       |
+----------+-------+----------+----------+
1 row in set (0.08 sec)
```

Maintenance des tables avec les outils shell

Depuis la version 3.23.38, un nouvel outil de vérification et de réparation des tables MyISAM a été mis en place : **mysqlcheck**. La différence avec myisamchk est

qu'il peut être utilisé alors que le serveur est en cours d'exécution, ce qui n'était pas possible avec myisamchk.

mysqlcheck permet de vérifier, analyser, optimiser et réparer les tables. Il y a trois manières de l'utiliser :

```
shell> mysqlcheck [OPTIONS] nom_base_données [tables]
shell> mysqlcheck [OPTIONS] --databases DB1 [DB2 DB3...]
shell> mysqlcheck [OPTIONS] --all-databases
```

Vérifier quelles options sont supportées par la version mysqlcheck installé avec la commande `mysqlcheck --help`. En voici quelques-unes :

--auto-repair
Lors d'une opération de vérification des tables, chaque table corrompue sera automatiquement réparée. La réparation a lieu une fois que toutes les tables ont été vérifiées.

-a ou --analyze
Analyse les tables.

-c ou --check
Vérifie que les tables ne comportent pas d'erreur.

-C ou --check-only-changed
Vérifie seulement les tables qui ont été modifiées depuis la dernière vérification.

-B, --databases
Vérifie plusieurs bases de données.

-F ou --fast
Vérifie seulement les tables qui ont été fermées correctement.

-h ou --host=...
Nom de domaine ou adresse IP du serveur auquel se connecter.

-o, --optimize
Optimiser la table, équivalent à OPTIMIZE TABLE

-p, --password[**=...**]
mot de passe de l'utilisateur mysql utilisé pour se connecter

-P, --port=... Numéro de port du serveur, dans le cas d'une connexion TCP/IP

-r, --repair Peut résoudre quasiment tous les problèmes de clés qui devraient être uniques mais qui ne le sont pas.

-S, --socket=... Fichier de socket à utiliser pour se connecter au serveur.

--tables Tables à vérifier, annule l'option `--databases`.

-u, --user=... Utilisateur de connexion au serveur MySQL.

5.1.3 MERGE

Ce type de table est disponible depuis la version MySQL-3.23.25. Une table de type MERGE, encore appelé MRG_MyISAM, est l'assemblage de plusieurs tables identiques.

Un fichier .MRG contient quant à lui la liste des fichiers d'index (.MYI). Toutes les tables utilisées doivent être dans la même base de données que la table MERGE.

Ce type de table peut aider à résoudre des problèmes comme :
- Une gestion plus facile de table destinée à contenir des logs. Par exemple, on peut séparer les logs de chaque mois dans des fichiers séparés, et les utiliser comme une seule table MERGE.
- La réparation de plusieurs petites tables est plus facile que la réparation d'une seule grosse.

Pour créer ce type de table on peut seulement utiliser des tables identiques de type MyISAM.

Quand on crée une table de type MERGE, il faut définir avec **UNION(liste-de-tables)** quelles tables on veut utiliser comme une seule. Avec l'option **INSERT_METHOD** on précise dans quelle table l'insertion doit commencer. Cette option peut prendre les valeurs **LAST** pour la dernière table dans la liste UNION, **FIRST** pour commencer par la première table de la liste.

L'exemple ci-dessous nous montre comment créer une table 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 ("Test"),("table"),("t1");
INSERT INTO t2 (message) VALUES ("Test"),("table"),("t2");
CREATE TABLE total (a INT NOT NULL, message CHAR(20), KEY(a))
             TYPE=MERGE UNION=(t1,t2) INSERT_METHOD=LAST;
```

A noter que nous n'avons pas créé de clé primaire dans la table `total` car elle ne sera pas unique.

5.1.4 HEAP

Présentation

Les tables HEAP sont entièrement stockées en mémoire. Cela les rend extrêment rapide, mais si MySQL est arrêté, toutes les données sont perdues. Ce type de table est très pratique pour les tables temporaires.

Création

La création d'une table HEAP se fait de cette manière :

```
mysql> CREATE TABLE nom_table (champ1 type, champ2 type ...) TYPE=HEAP
max_rows=nombre_lignes
```

Informations à prendre en considération

Il est fort conseillé de définir le nombre de lignes maximum, de manière à ne pas occuper toute la mémoire avec la seule table.

Les tables HEAP ne supportent pas les champs de type `blob, text, auto_increment`

Les tables HEAP ne supporte pas d'index sur un champ pouvant être égal à NULL.

Il est possible d'avoir plusieurs clés identiques.

Les recherches se font sur les clés en entier, pas comme avec les tables MyISAM où la recherche peut se faire sur une partie de clé.

Pour libérer de la mémoire, il faut soit vider les tables HEAP, soit les supprimer.

Options de mysqld

La variable **max_heap_table_size** permet de limiter la taille des tables HEAP.

5.1.5 InnoDB

Présentation

Le type de tables InnoDB apporte de nombreuses fonctionnalités encore manquantes à MySQL : les tables InnoDB sont transactionnelles, c'est à dire qu'il est possible de valider ou annuler les dernières modifications effectuées.
Il apporte également le support pour les clés étrangères.
Les données sont stockées dans ce que l'on appelle un **tablespace**, il s'agit en fait d'un fichier contenant les tables et les index. Il est possible d'en avoir plusieurs.
Les verrous sont posés sur les lignes et non sur les tables entières comme c'est le cas avec les tables MyISAM, ce qui permet une plus grande concurrence, c'est-à-dire qu'un plus grand nombre d'utilisateurs peuvent travailler en même temps.
Il existe un réél outil de sauvegarde "à chaud". C'est à dire qu'il est possible de réaliser la sauvegarde d'une base en cours d'utilisation sans poser de verrous sur les tables. C'est outil est propriétaire.

Toutes les dernières informations à propos d'InnoDB sont disponibles sur http://www.innodb.com. InnoDB est distribué sous licence GNU GPL.

Ajout du support InnoDB

Pour avoir le support InnoDB, il faut avoir compilé MySQL avec l'option de configuration `--with-innoDB` ou avoir installé le paquetage MySQL-Max.

Et il faut également dans le fichier **my.cnf** au moins la ligne suivante :
`innodb_data_file_path=chemin_données:30M ;chemin_données2:40M`,

qui permet d'indiquer le chemin vers les fichiers de données et leur taille. La somme de la taille de tous les fichiers doit être au moins égale à 10M.

Mais il est préférable, pour obtenir de meilleures performances, de spécifer d'autres options :

innodb_data_home_dir : répertoire de base des données InnoDB. Quand cette variable est spécifiée, le chemin absolu des fichiers de données est obtenu en concaténant le contenu de `innodb_data_home_dir` avec le chemin de chaque fichier spécifié dans `innodb_data_file_path`. Si cette variable n'est pas spécifiée, le répertoire sera le répertoire *DATA-DIR* de MySQL.

innodb_log_group_home_dir : répertoire des fichiers de log.

innodb_log_files_in_group : nombre de fichiers de log utilisés. InnoDB écrit dans les fichiers de log de manière tournante. Il est conseillé de mettre cette variable à 3.

innodb_log_file_size : taille de chaque fichier de log en Mo. La somme de la taille de tous les fichiers de log doit être inférieure à 4Go sur les systèmes 32 bits

innodb_log_buffer_size : taille des buffers utilisés par InnoDB pour écrire les fichiers de log sur le disque.

innodb_buffer_pool_size : taille de la mémoire utilisée pour stocker des données et des index en cache. Plus ce buffer sera grand, moins il y aura d'accès disque. Pour un serveur dédié aux bases de données, il est possible que ce paramètre vale jusqu'à 80% de la taille de la mémoire physique.

innodb_flush_log_at_trx_commit : donner la valeur 1 à ce paramètre signifie qu'à chaque validation de transaction, les logs correspondants sont écrits dans les fichiers de log, de manière à ce que ces modifications deviennent permanentes, même après un plantage de la base de données. Lui donner la valeur 0, dans le cas où il y a très peu de transactions, permet de réduire les accès disque.

Exemple de section [mysqld] du fichier my.cnf pour un serveur supportant InnoDB :

```
[mysqld]
...
#les données sont stockées dans les fichiers /opt/mysql/var/ibdata1 et
# /opt/mysqlvar/ibdata2. Il est possible de mettre l'option autoextend sur
#tant le dernier fichier de données, qui l'autorise à \^etre plus gros que 300M
#qu'il ne dépasse pas la valeur maximum de 2Go
innodb_data_home_dir = /opt/mysql/var/
innodb_data_file_path =ibdata1:400M;ibdata2:300M:autoextend:max:2000M

#les fichiers de log InnoDB se trouvent dans /opt/mysql/var
# et sont au nombre de trois
innodb_log_group_home_dir = /opt/mysql/var/
set-variable = innodb_mirrored_log_groups=1
set-variable = innodb_log_files_in_group=3

# la taille de chaque fichier de log est de 5Mo
set-variable = innodb_log_file_size=5M
set-variable = innodb_log_buffer_size=8M
innodb_flush_log_at_trx_commit=1
set-variable = innodb_buffer_pool_size=16M
```

Création d'une table InnoDB

Lors de la création de la table, il suffit de préciser quel type de table on veut utiliser avec la directive **type** :

```
mysql>CREATE TABLE CUSTOMER (A INT, B CHAR (20), INDEX (A)) TYPE = InnoDB;
```

Cette commande crée une table nommée CUSTOMER et un index sur la colonne A dans le tablespace défini dans le fichier *my.cnf*. Dans le même temps, InnoDB ajoutera une entrée à son dictionnaire de données pour la table **base_courante/CUSTOMER**, de cette manière plusieurs bases peuvent contenir une table de même nom.

Les clés étrangères

Le support pour les clès étrangères est disponible depuis la version 3.23.43b de MySQL. La définition d'une clé étrangère se fait de cette manière :

```
FOREIGN KEY (nom_col_index, ...)
                  REFERENCES nom_table (nom_col_index, ...)
                  [ON DELETE CASCADE | ON DELETE SET NULL]
```

Toutes les tables doivent être de type InnoDB, et il doit y avoir un index sur les clés étrangères. InnoDB ne crée pas automatiquement d'index sur les clés étrangères, il faut les créer explicitement.

Une clé étrangère doit être de même format que la clé à laquelle elle fait référence : même taille, le fait qu'un entier soit signé ou non, ainsi que la longueur d'un champ de type chaîne de caractères.

Les options **ON DELETE CASCADE** et **ON DELETE SET NULL** sont disponibles depuis la version MySQL-3.23.50. Si la première est activée, lorsqu'un enregistrement de la table "mère" est effacée, tous les enregistrements de la table "fille", dont la clé étrangère est égale à la clé de la table mère, sont effacés. Si c'est la deuxième qui est activée, le champ sera remplacé par la valeur NULL dans la table "fille".

Exemple : créer une table nommée **parent** et une table nommée **enfant** contenant une clé étrangère faisant référence à la clé de la table parent.

```
mysql>CREATE TABLE parent(id INT NOT NULL, PRIMARY KEY (id)) TYPE=INNODB;
mysql>CREATE TABLE enfant(id INT, parent_id INT, INDEX par_ind (parent_id),
             FOREIGN KEY (parent_id) REFERENCES parent(id)
             ON DELETE SET NULL
) TYPE=INNODB;
```

En exécutant la commande **show table status ;** on obtient les informations sur les clés étrangères des différentes tables de la base de données, ainsi que l'espace encore disponible pour les tables InnoDB.

Dans la colonne "Comment" renvoyée par la commande **show table status**, des informations sur l'utilisation des tablespace et les clés étrangères sont fournies.

```
mysql> show table status;
+--------+...+-------------------------------------------------------------------+
| Name   |   | Comment                                                           |
+--------+...+-------------------------------------------------------------------+
| enfant |   | InnoDB free: 3819 kB; (parent_id) REFER nom_base/parent(id)   |
| parent |   | InnoDB free: 3819                                                 |
+--------+...+-------------------------------------------------------------------+
```

Depuis la version MySQL-3.23.50, il est possible d'ajouter une clé étrangère grâce à la commande ALTER :

```
mysql> ALTER TABLE nom_table_fille
         ADD CONSTRAINT FOREIGN KEY (...) REFERENCES nom_table_mère(...)
```

Ne pas oublier de créer un index avant de créer la clé étrangère.

Le modèle transactionnel d'InnoDB

Contrairement aux autres types de tables MySQL, où les verrous sont posés sur la table entière, avec le type InnoDB, les verrous sont posés uniquement sur la (ou les) lignes concernant la transaction en cours. Ceci permet un meilleur accès concurrentiel à la base de données.

Avec InnoDB, toutes les requêtes sont effectuées dans des transactions. Si MySQL est en mode **auto-commit**, chaque requête correspondra à une transaction. En revanche, si le mode `auto-commit` est désactivé, les modifications réalisées pendant la transactions ne prendront réellement effet, et ne seront visibles par tout le monde, qu'après un **COMMIT**. Elles peuvent aussi être annulées grâce à un **ROLLBACK**.

Lecture consistante : c'est le mode utilisé par InnoDB pour fournir une capture de la base de données à un moment précis. Quand une transaction commence, elle effectue une première lecture consistante. C'est cette lecture qui sera utilisée tout au long de son existence. Tous les changements effectués, et validés par les autres transactions après le commencement de la transaction ne seront pas visibles par celle-ci. Seules les modifications effectuées par les requêtes de la transaction en cours seront visibles. La lecture consistante est le mode par défaut qu'InnoDB utilise pour exécuter des requêtes **SELECT**. Une requête SELECT ne pose pas de verrou, donc les autres utilisateurs peuvent modifier les données en même temps qu'une lecture consistante a lieu.

Exemple de lecture consistante avec InnoDB : La commande `set autocommit=0` permet de désactiver le mode autocommit, et de passer réellément en mode transactionnel. On voit bien que l'utilisateur A n'a pas accès aux modifications apportées par l'utilisateur B sur la table **t** même après que celui a validé

ses changements par un COMMIT. Tout le long de la transaction, l'utilisateur voit le contenu qu'il y avait au moment de la lecture consistante effectuée au début de la transaction.

```
          Utilisateur A           Utilisateur B

         SET AUTOCOMMIT=0;        SET AUTOCOMMIT=0;
temps
|        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    |
         ---------------------
```

Il est possible néanmoins pour l'utilisateur A de voir les modifications validées par B, avant d'exécuter son commit. Pour cela A doit utiliser l'option suivante :

```
mysql>select * from t LOCK IN SHARE MODE;
```

Restriction sur les tables InnoDB

Il faut être prudent quant à l'utilisation que l'on fait des tables InnoDB. Il existe certaines choses qu'on ne peut pas faire :

- Il ne faut pas convertir les tables de la base **mysql** de MyISAM vers InnoDB. Ce n'est pas supporté. Si jamais une conversion était faite, MySQL ne redémarrerait pas tant que les tables ne seraient pas restaurées à partir d'une sauvegarde ou regénérée avec le script **mysql_install_db**.
- On ne peut pas créer de clé sur des champs de type *blob* ou *text*.
- On ne peut pas créer un index de type *unique* sur une partie d'une champ, comme dans l'exemple ci-dessous :

```
mysql>CREATE TABLE T (A CHAR(20), B INT, UNIQUE (A(5))) TYPE = InnoDB;
ERROR 1089: Mauvaise sous-clef. Ce n'est pas une 'string' ou la longueur
dépasse celle définie dans la clef
```

5.1.6 BerkeleyDB

Présentation

Le support pour les tables de type **BerkeleyDB** ou **BDB** est disponible depuis la version MySQL-3.23.34.

Les tables BDB apportent elles aussi le mode transactionnel à MySQL, ainsi que le **COMMIT** et **ROLLBACK** sur les transactions. Ce type de table ne présente pas de réél intérêt, comparé au type InnoDB, et le support proposé par MySQL pour l'exploiter demande encore à être optimisé. Nous ne nous attarderons donc pas sur ce type de table.

Pour de plus amples informations, consulter le site officiel de BerkeleyDB à l'adresse suivante : http://www.sleepycat.com/.

5.2 La réplication

Comment augmenter la disponibilité d'un serveur MySQL ?

5.2.1 Présentation

La réplication est un mécanisme avancé que l'on retrouve sur la majorité des bases de données de production. Elle permet de cloner un serveur de bases de données, que l'on appelle serveur maître, vers d'autres serveurs que l'on appelle serveurs esclaves.

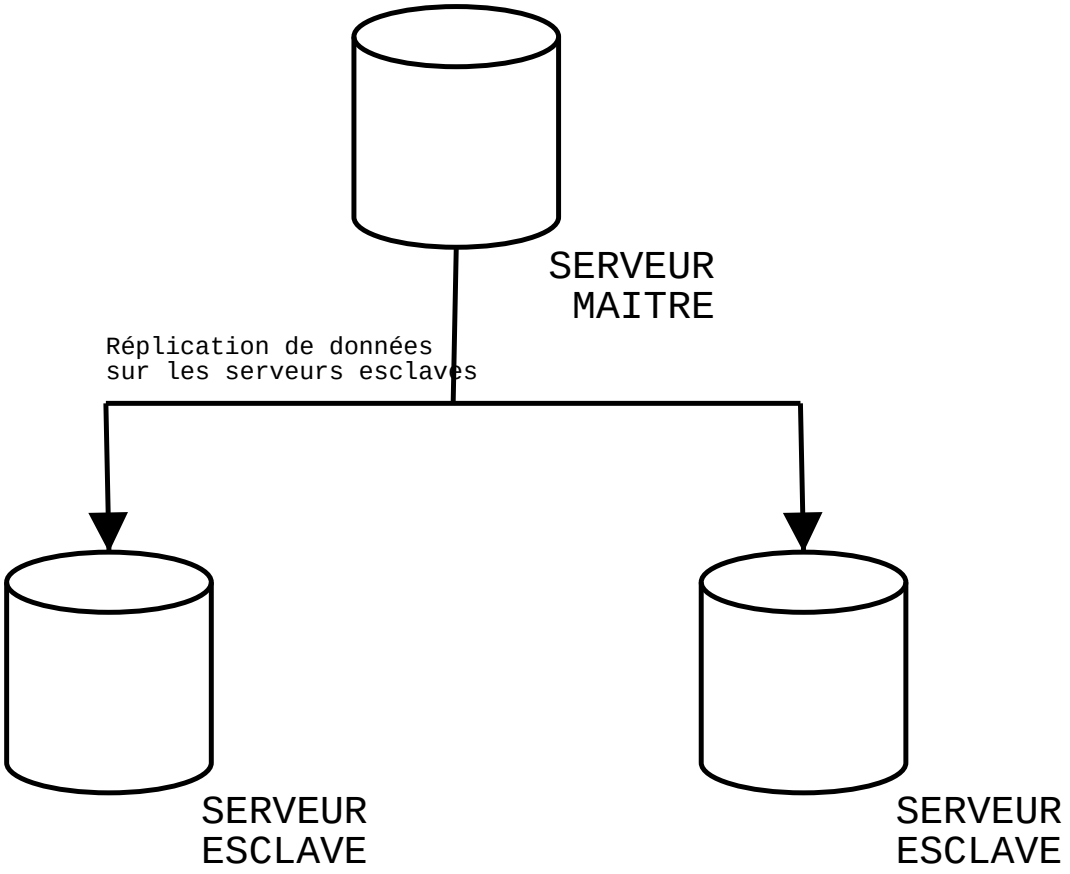


FIG. 5.1: *Réplication*

Elle apporte une solution aux problèmes de fiabilité et de disponibilité :

Fiabilité : les données sont sauvegardées sur différents serveurs. En cas d'indisponibilité de la base de données maître, il est possible de la remplacer en mettant en place un serveur esclave.

Disponibilité : les données sont présentes sur différents serveurs. Ainsi, il est possible d'alléger les tâches du serveur maître en les effectuant sur les différents serveurs esclaves.

Attention, il est important de comprendre les limitations de la réplication sous MySQL !

En effet, les écritures et les modifications effectuées sur le serveur maître sont répliquées sur les serveurs esclaves mais la réciproque est fausse. Toutes les modifications doivent donc s'effectuer sur le serveur maître. Dans la majorité des cas, ceci ne pose pas de problèmes considérables puisque les opérations de consultation représentent la partie la plus importante des requêtes ; toutefois dans le cas contraire, la réplication ne peut apporter de solution satisfaisante.

5.2.2 Mise en oeuvre

La mise en place de la réplication s'effectue par l'intermédiaire du fichier de configuration `my.cnf` de chaque serveur intervenant dans la réplication : le maître comme les esclaves.

Configuration du serveur maître

1. S'assurer que la version de MySQL est récente : supérieure à 3.23.29. Les versions précédentes utilisent des fichiers de log d'un format différent et comportent quelques bogues qui ont été corrigés par la suite.
2. Créer un utilisateur ayant le privilège FILE pour les versions de MySQL antérieures à 4.02, ou le privilège REPLICATION SLAVE pour les versions plus récentes. S'assurer aussi que l'utilisateur a le droit de se connecter depuis tous les esclaves. Il est recommandé de créer un utilisateur dédié à la réplication, et donc de ne pas lui accorder de droits supplémentaire à celui cité précédemment.
   Exemple : créer un utilisateur `replication` permettant aux esclaves de se synchroniser :
   ```
   mysql>GRANT FILE ON *.* TO replication@"%" IDENTIFIED BY 'mot_de_passe';
   ```
3. Arrêter le serveur maître :
   ```
   shell> mysqladmin -u root -p shutdown
   Enter password :
   ```
4. Effectuer une sauvegarde des bases de données avec la commande suivante :
   ```
   tar czvf /tmp/mysql-snapshot.tar.gz /MYSQL-DATA-DIR
   ```
5. Dans la section [**mysqld**] du fichier **my.cnf**, ajouter les directives suivantes :
   log-bin : active l'archivage des modifications sur le serveur maître dans des fichiers que les esclaves vont venir lire pour se synchroniser.
   server-id=nombre_unique : il est très important que ce numéro soit unique pour chaque serveur MySQL.
6. Redémarrer le serveur.

Il est possible de passer d'autres options au serveur mysqld :

binlog-do-db=nom_base : cette directive permet mettre dans les fichiers de log binaires toutes les modifications effectuées lorsque la base de données courante est *nom_base*. Avant de la mettre en place, s'assurer que les modifications effectuées lorsque la base courante est *nom_base* ne concernent pas d'autres bases de données.

binlog-ignore-db=nom_base : toutes les modifications effectuées lorsque la base courante est *nom_base* ne seront pas inscrites dans le fichier de log. Là encore, s'assurer que les changements ne concernent que la base de données courante.

Configuration d'un serveur esclave

1. ajouter les lignes suivantes dans la section [**mysqld**] du fichier **my.cnf** :

```
master-host = <nom de machine du serveur maître>
master-user = replication
master-password = <mot_de_passe>
server-id = <nombre_unique>
```

2. Copier le snapshot effectué sur le serveur maître dans le répertoire de données du serveur MySQL esclave.

```
tar xzvf /tmp/mysql-snapshot.tar.gz -C /MYSQL-DATA-DIR
```

3. Redémarrer le serveur MySQL esclave. Un fichier **master.info** est généré dans le répertoire des données de l'esclave. Ce fichier contient les informations sur le serveur maître nécessaire à la réplication.

Les options suivantes, à ajouter dans la section [mysqld], sont également disponibles :

master-port=numero_port : port de connexion TCP/IP du serveur maître

master-connect-retry=secondes : nombre de secondes avant que l'esclave ne retente une connexion au maître en cas de problème.

replicate-do-table=nom_base.nom_table : dit au serveur de restreindre le réplication à la table mentionnée. Pour répliquer plusieurs tables, ajouter la directive plusieurs fois dans dans le fichier *my.cnf*, une fois pour chaque table.

replicate-ignore-table=nom_base.nom_table : dit au serveur de ne pas répliquer la table mentionnée. Cette directive peut aussi être ajoutée plusieurs fois dans le fichier *my.cnf*, pour ignorer plusieurs tables.

replicate-ignore-db=nom_base : ignorer la réplication de la base de données passée en paramètre. Pour ignorer plusieurs bases, ajouter la directive pour chacune d'entre elles. Cette option ne fonctionne pas dans le cas où l'on procède à des mises à jour croisées de bases de données.

replicate-do-db=nom_base : répliquer uniquement la base de données citée. Il est possible d'utiliser la directive plusieurs fois : une fois pour chaque base de données.

replicate-rewrite-db=nom_source− >nom_destination : Répliquer la base de données ayant le nom *nom_source* sur le serveur maître vers la base ayant le nom *nom_destination* sur le serveur esclave.

slave-skip-errors=[code_err1,code_err2,... — all :] cette option est disponible depuis la version 3.23.49 de MySQL. Dit au serveur esclave de continuer le réplication lorsqu'une requête renvoie un des codes d'erreur listés dans cette directive. N'utiliser cette option uniquement si on connait l'origine de ces erreurs. L'option `all` n'est bien sûr pas recommandée, en effet l'intégrité des données risque d'être perdue.

skip-slave-start : dit au serveur esclave de ne pas commencer la réplication au démarrage du serveur. Il faudra utiliser la commande **SLAVE START** pour la lancer au moment voulu.

slave_net_timeout=secondes : nombre de secondes à attendre des données du serveur maître avant d'abandonner la lecture.

A partir de la version 4.0.0 de MySQL il sera possible de répliquer via ssl. Utiliser les directives suivantes pour l'activer :

master-ssl : active le protocle SSL pour la réplication.

master-ssl-key : nom du fichier contenant la clé du serveur maître. Ne fonctionne que si la directive master-ssl a été activée.

master-ssl-cert : nom du fichier de certificat du serveur maître. Ne fonctionne que si la directive master-ssl a été activée.

5.2.3 Les commandes relatives à la réplication

Pour assurer la maintenance et l'administration de la réplication, plusieurs commandes, à exécuter dans une interface SQL, sont mises à notre disposition.

Sur le serveur maître

set sql_log_bin=[0—1 :] active ou désactive la mise à jour des fichiers de log binaires lus par les esclaves pour se synchroniser. L'utilisateur doit avoir le privilège *PROCESS* pour pouvoir exécuter cette commande.

reset master : efface tous les fichiers de log binaires listés dans le fichier d'index .

show master status : renvoie les informations sur l'état du serveur maître : dans quel fichier de log il est en train d'écrire, et sa position dans celui-ci.

show master logs : renvoie la liste des fichiers de log binaires.

purge master logs to 'fichier_log' : purge les fichiers de log binaire. Le fichier de log passé en paramètre devient alors le premier dans la liste. Si jamais un esclave est en train de lire un des fichiers de log, la commande ne fera rien et un message d'erreur sera renvoyé. Vérifier d'abord avec les commandes *show slave status* et *show master logs* pour voir s'il est possible d'exécuter celle-ci.

show slave hosts : Disponible à partir de la version 4.0.0 de MySQL. Renvoie la liste des esclaves connectés au serveur.

Sur le serveur esclave

slave start : démarrer la réplication sur le serveur esclave.

slave stop : arrête la réplication sur le serveur esclave.

set sql_slave_skip_counter=n : ne pas effectuer les n prochaines requêtes. Cette commande ne peut être exécutée que si l'esclave n'est pas en train de répliquer. S'il est en cours d'exécution, un message d'erreur sera renvoyé. Cette option peut être utile pour résoudre un problème de réplication. Mais il faut être sûr que la requête qui sera ignorée ne risque pas d'altérer l'intégrité des données.

reset slave : force l'esclave à oublier sa position de réplication dans les fichiers de log du maître.

load table nom_table from master : télécharger une copie de la table fournie en paramètre sur l'esclave.

change master to liste_def_maitre : modifie les informations concernant le serveur maître et relance la réplication. La liste de définition est la suivante :

```
CHANGE MASTER TO
  MASTER_HOST='maitre2.com',
  MASTER_USER='replication',
  MASTER_PASSWORD='mot\_de\_passe',
  MASTER_PORT=3306,
  MASTER_LOG_FILE='maitre2-bin.001',
  MASTER_LOG_POS=4;
```

Seules les valeurs qui sont modifiées ont besoin d'être passées en paramètres, toutes les valeurs qui ne sont pas précisées resteront inchangées. A noter que lorsque le serveur sera redémarré, les informations de l'ancien serveur seront réutilisées. Pour éviter cela, effacer le fichier **master.info**, de manière à ce que le serveur relise le fichier **my.cnf**.

show slave status : renvoie des informations sur l'état de réplication de l'esclave : adresse du serveur maître, position dans les fichiers de log, réplication en cours d'exécution ou non.

Attention, les tables de type InnoDB deviennent de type MyISAM sur les serveurs esclaves. Cela dit, dans la mesure où aucune modification ne doit être réalisée sur les serveurs esclaves, ce n'est pas réellement gênant : l'intégrité des données est assurée sur le serveur maître.

Origine d'un problème de réplication

Il arrive parfois que la réplication ne se fasse plus sur un serveur esclave. Pour connaître l'origine du problème, la marche à suivre est la suivante :

1. D'abord vérifier si le serveur maître et le serveur esclave en sont au même point dans les fichiers de log, en exécutant **show master status** et **show slave status**. Quand tout va bien, les serveurs maître et esclaves sont dans le même fichier de binaire et à la même position.

2. Si sur l'esclave, la colonne "Slave Running" est à "No", un message d'erreur est inscrit dans la colonne "Last Error". S'il n'y a pas de message d'erreur, cela signifie certainement que l'esclave a été arrêté normalement, et que la commande **slave start** suffit.
3. Si la réplication ne redémarre pas et qu'il n'y a pas de message d'erreur dans la colonne "Last Error", consulter le fichier *hostname.err* situé dans le dossier *DATA-DIR* du serveur esclave.
4. Dans le cas où la commande *show slave status* renvoie un message d'erreur dans la colonne "Last Error", et que la requête qui pose problème peut être abandonnée, faire : **slave stop ;** puis **set sql_slave_skip_counter=1** et **slave start**. Vérifier que la réplication est bien repartie avec **show slave status**.

L'utilitaire **mysqlbinlog** permet lire le contenu des fichiers de log binaires, de manière à trouver quelle requête pose problème.

Sa syntaxe :

```
shell > mysqlbinlog [OPTIONS] fichier_log_binaire
```

Parmi les options de mysqlbinlog, on a :

-o nombre : définir l'offset. N'affiche pas les *nombre* entrées.

- s : n'affiche que les requêtes, sans les informations autour.

-h serveur_hôte : lit les fichiers de log du serveur passé en paramètre à cette option.

-P numero_port : port de connexion au serveur.

-u nom : nom d'utilisateur de connexion.

-p mot_de_passe : mot de passe de l'utilisateur.

-j position : aller directement à la position mentionnée dans le fichier.

5.2.4 Mises en garde : problèmes connus

Quelques situations peuvent poser problème lors de la réplication :

- La fonction RAND() ne se réplique pas correctement.
- Il faut absolument que les serveur maître et esclave soient configurés avec le même jeu de caractères (directive `--default-character-set`), de manière à éviter les erreurs de duplication de clé sur les esclaves. Deux valeurs différentes peuvent être vues comme identiques avec deux jeux de caractères différents.
- Une requête du type LOAD DATA INFILE se répliquera correctement si le fichier est encore sur le serveur maître au moment de la réplication.
- Une commande LOAD DATA LOCAL INFILE sera ignorée. A partir de la version 4.0 de MySQL cette directive est correctement répliquée.
- Les commandes FLUSH ne sont pas stockées dans les fichiers de log binaires et ne sont donc pas répliquées non plus. Ca ne pose aucun problème sauf si on réplique la base *mysql* et qu'on ne modifie pas les droits avec la commande GRANT mais en manipulant directement les données, qui nécessite d'exécuter la commande FLUSH PRIVILEGES. A ce moment là il faudra exécuter à la main sur chaque serveur esclave la commande FLUSH PRIVILEGES.

ATTENTION ! ! ! Il est important de se souvenir que la réplication MySQL est unidirectionnelle. C'est à dire que les modifications ne peuvent être apportées que depuis le serveur maître. Si ce n'est pas le cas, on peut se retrouver rapidement face à des problèmes de clé primaire dupliquée ou d'intégrité de données.

5.3 De la bonne utilisation des index

Qu'est-ce qu'un index ? Comment l'utiliser ?)

5.3.1 Rôle d'un index

Les index sont utilisés pour trouver un enregistrement dans une table très rapidement.

Sans index, un SGBD serait obligé de chercher dans la table de manière séquentielle, en commençant par la première, jusqu'à celle correspondant aux conditions définies dans la requête. Dans ce cas là, plus la table est grosse et plus la recherche est longue.

Avec des index, le SGBD peut situer rapidement dans la table l'enregistrement qu'il recherche. Si une table contient 1000 enregistrements, une recherche avec des index est environ 100 fois plus rapide qu'une recherche séquentielle. A noter tout de même que si la recherche concerne la quasi-totalité des 1000 enregistrements, il est plus rapide de faire une lecture séquentielle, et cela évite de nombreux accès disque.

Les index sont utilisés pour :
- trouver rapidement les lignes correspondant à la clause WHERE.
- extraire des lignes d'autres tables lors de la création de jointure.
- trouver le MAX() ou le MIN() d'une colonne.

Pour que MySQL utilise les index, le préfixe de l'index doit être utilisé dans chaque groupe AND de la clause WHERE. Le préfixe, c'est la partie gauche de l'index.

Exemple : la table **test** possède un index formé sur trois colonnes : **col1, col2 et col3**. La requête suivante utilisera l'index :

```
mysql>select * from test where col1=valeur1 and col2=valeur2
      -> or col1=valeur1 and col3=valeur2;
```

En revanche celle-ci n'utilisera pas l'index, car la clause where ne contient pas la partie de gauche de l'index :

```
mysql>select * from test where col2=valeur1 and col3=valeur2;
```

5.3.2 Les différents types d'index

Tous les types de données de MySQL peuvent être indexés.

Pour les champs de type `CHAR` ou `VARCHAR`, il est possible d'indexer un préfixe de la colonne. C'est plus rapide et nécessite moins d'espace disque.
Exemple : création d'un index sur les 10 premiers caractères du champ `nom` :

```
mysql> CREATE TABLE test (
    -> nom CHAR(200) NOT NULL,
    -> KEY nom_index (nom(10)));
```

Pour les champs de type `BLOB` ou `TEXT`, les index sont obligatoirement créés sur un préfixe de la colonne. Il est impossible d'indexer la colonne entière.

Les index sur plusieurs colonnes peuvent être considérés comme étant la concaténation de toutes les colonnes. Il est possible d'utiliser jusqu'à 15 colonnes pour créer un index. **Exemple :** création d'une table possédant un index sur plusieurs colonnes :

```
mysql> CREATE TABLE test (
    -> id INT NOT NULL,
    -> nom_famille CHAR(30) NOT NULL,
    -> prenom CHAR(30) NOT NULL,
    -> PRIMARY KEY (id),
    -> INDEX nom (nom_famille,prenom));
```

Dans ce cas l'index **nom** sera utilisé uniquement si la clause where contient le préfixe de cet index dans chacun de ses groupes `AND`. Le préfixe étant le champ **nom_famille**.

Exemple : cette requête fait appel à l'index :

```
mysql> SELECT * FROM test WHERE nom_famille="Dupont";
```

tandis que celle-ci n'utilise pas l'index :

```
mysql> SELECT * FROM test WHERE prenom="Lionel";
```

5.3.3 Quand créer un index ?

Créer un index sur une table peut avoir l'effet inverse à celui souhaîté si les opérations réalisées sur cette table ne s'y prêtent pas. Il est intéressant de créer un index dans les cas suivants :

- Si les requêtes effectuées sur la table portent sur moins de 30% des données de la table.
- Si les colonnes sont utilisées dans des jointures, un index peut sensiblement améliorer les performances.
- Les colonnes dont les valeurs sont uniques ou très dispersées.
- Dans le cas de l'utilisation de tables InnoDB, ne pas oublier de créer un index sur les clés étrangères.

Eviter de créer des index dans les cas suivants :

- Les colonnes comportent peu de valeurs distinctes.
- La table ne contient qu'un faible volume de données.

Atelier 5 : Optimisation et administration avancée de MySQL

Exercice 5.1 *Activation du support InnoDB*

En modifiant le fichier my.cnf, activez le support InnoDB. Faites en sortes que les fichiers de données soient stockées dans le répertoire */var/lib/mysql/innodb*. Redémarrez le serveur MySQL et vérifiez que le support est bien activé en créant une table, puis en vérifiant son type avec la commande **show table status ...**

Exercice 5.2 *Réalisation de la base de données gestion de commandes*

1. En utilisant des tables de type InnoDB, réalisez le Modèle Physique de Données correspondant au MLD de l'exercice "gestion de commandes".
2. Exécutez la requête **show table status** et vérifiez que toutes les tables sont bien de type InnoDB et que les clés étrangères ont été correctement déclarées.

Exercice 5.3 *Mise en oeuvre de la réplication*

1. Mettre en place la réplication sur la base "gestion de commandes" avec votre voisin, l'un étant le serveur maître, l'autre l'esclave.
2. Vérifiez qu'elle fonctionne en modifiant le contenu de la base sur le serveur maître puis en vous assurant que les modifications ont bien été reproduites sur les esclaves.

6

Outils d'administration graphiques

Objectifs

- Présentation de deux outils graphiques permettant de rendre l'administration de MySQL plus confortable.

Contenu

6.1 phpMyAdmin : aministration via HTTP 85
6.2 MySQL Control Center 90
Atelier 6 95

Références

- Le site de MySQL : http://www.mysql.com
- Le site de phpMyAdImin : http://www.phpmyadmin.net

6.1 phpMyAdmin : aministration via HTTP

Comment administrer son serveur MySQL via une interface web ?

Il existe de nombreux outils de gestion pour un serveur MySQL. Le plus connu est certainement phpMyAdmin. C'est une application web qui permet de tout gérer depuis son navigateur. Vous trouverez toutes les informations et la dernière version de l'application sur le site http://www.phpmyadmin.net/.

6.1.1 Installation

L'installation est simple : une fois l'archive téléchargée, commencer par la décompresser dans un répertoire accessible par Apache, par exemple */var/www/html*.

Ceci fait, on doit créer un utilisateur MySQL pour phpMyAdmin. Cet utilisateur ne sert qu'à consulter la liste des tables et aura donc des droits très limités.

Dans notre exemple, il s'appelle **phpma**.

```
$ mysql -u root -p
mysql> grant select on mysql.user to phpma@'%' identified by 'phpma';
mysql> grant select on mysql.db to phpma@'%';
```

Il faut maintenant configurer phpMyAdmin. Pour cela modifier le fichier *config.inc.php* qui se trouve dans le répertoire de l'application.

Les paramètres à modifier sont les suivants :

```
// Chemin d'installation de l'application
$cfgPmaAbsoluteUri = '/chemin/web/vers/phpmyadmin';

// Connection
$cfgServers[$i]['host']          = 'localhost';
$cfgServers[$i]['connect_type']  = 'tcp';

// Utilisateur MySQL pour phpMyAdmin
$cfgServers[$i]['controluser']   = 'phpma';
$cfgServers[$i]['controlpass']   = 'phpma';

// Type d'authentification
$cfgServers[$i]['auth_type']     = 'cookie';
```

PhpMyAdmin est maintenant installé, vous pouvez y accéder depuis votre navigateur.

6.1.2 Connexion

La connexion se fait donc grâce à un navigateur web :

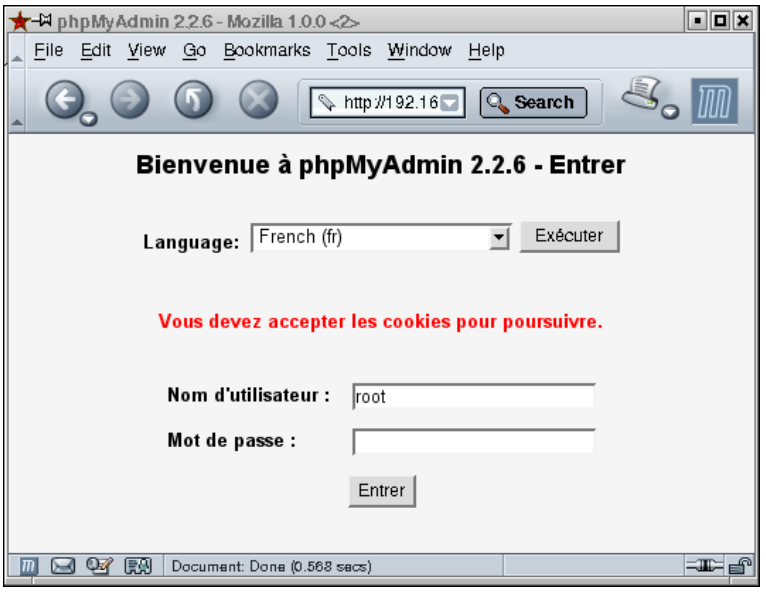


FIG. 6.1: *Authentification à phpMyAdmin*

Une fois connecté, la page d'accueil est la suivante :

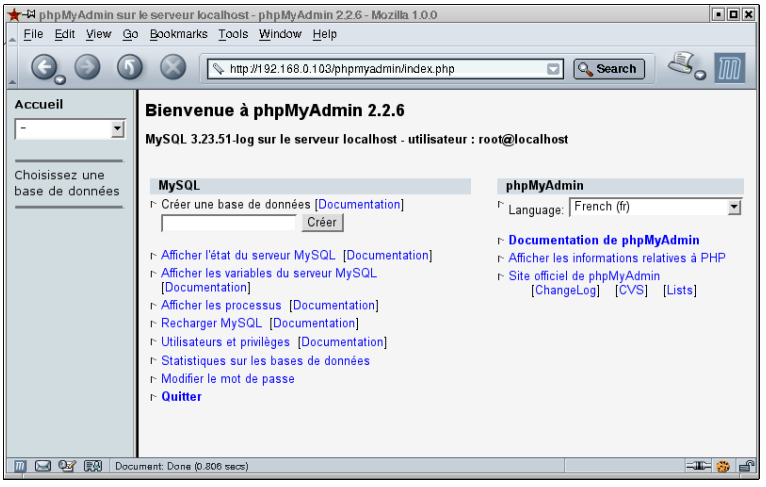


FIG. 6.2: *Page d'accueil de phpMyAdmin*

Plusieurs tâches d'administration sont alors accessibles : l'affichage des variables, gestion des processus (threads) en cours, la gestion des utilisateurs, l'affichage de statistiques concernant les bases de données. L'exemple ci-dessous nous affiche les statistiques concernant les différentes bases de données :

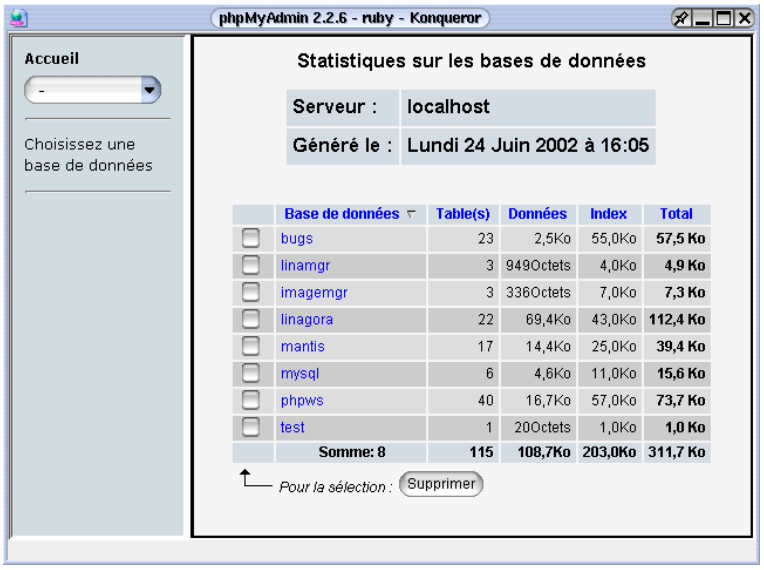


FIG. 6.3: *Statistiques*

6.1.3 Gestion des utilisateurs

Ci-dessous la liste de tous les utilisateurs de toutes les bases de données confondues, la colonne privilèges correspond aux privilèges globaux.

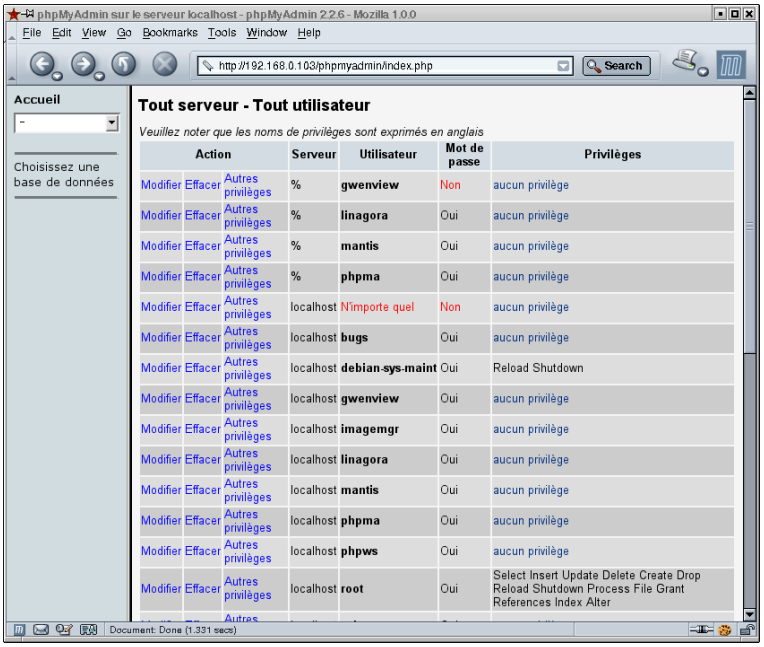


FIG. 6.4: *Liste des utilisateurs*

Un formulaire est disponible pour ajouter un utilisateur MySQL :

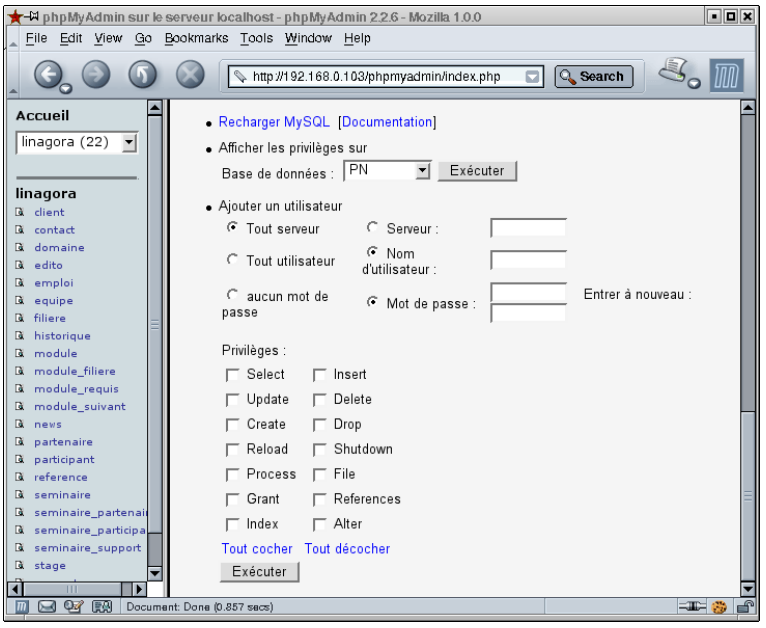


FIG. 6.5: *Ajout d'un utilisateur*

Equivalent à la commande shell **mysqlaccess**, cette page nous informe quant aux privilèges attribués sur la base de données sélectionnée.

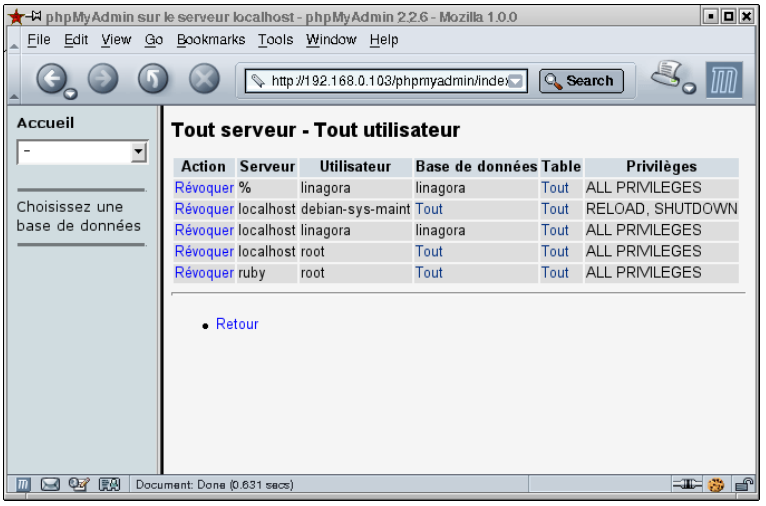


FIG. 6.6: *Liste des privilèges attribués sur la base linagora*

6.1.4 Gestion du contenu

L'interface phpMyAdmin permet aussi d'ajouter ou de supprimer des tables, de modifier leur structure, de manipuler les données ...

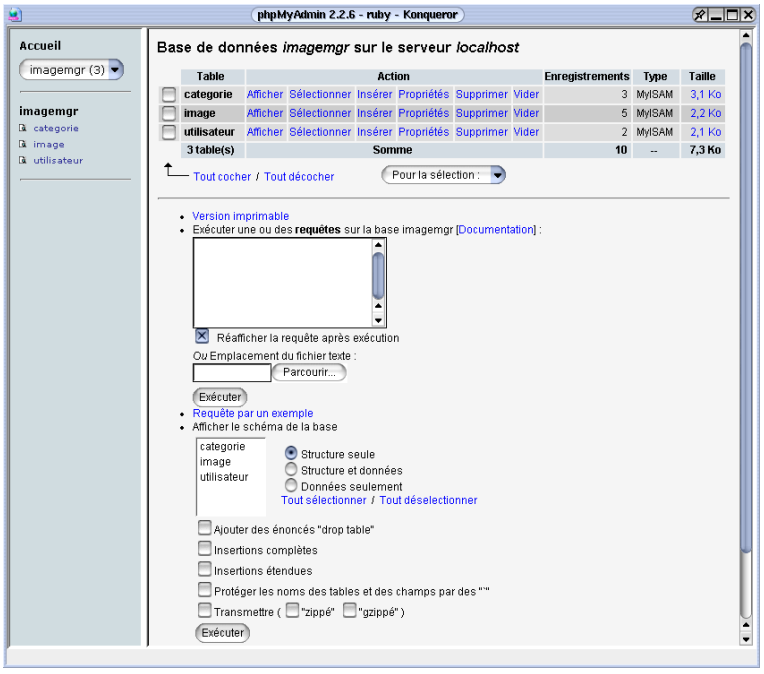


FIG. 6.7: *Contenu d'une base de données*

La page suivante correspond à l'affichage du contenu d'une table :

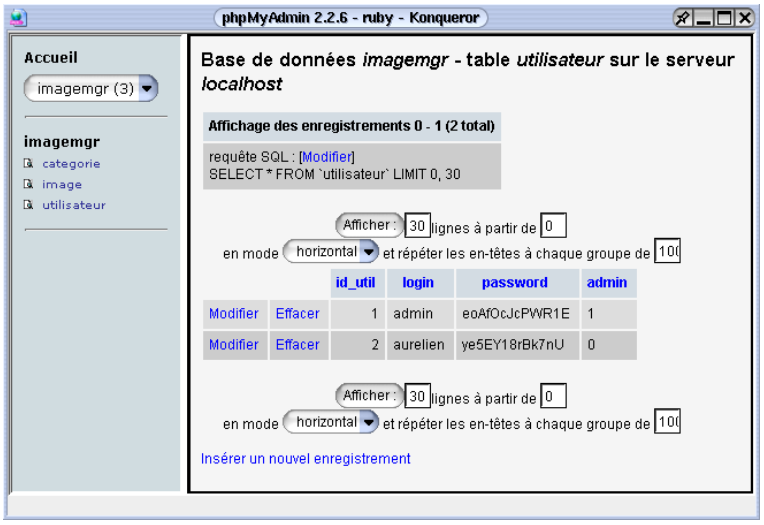


FIG. 6.8: *Contenu d'une table*

Il est possible de modifier ou de supprimer chaque enregistrement de la table, et également d'en ajouter.

6.2 MySQL Control Center

Un client MySQL graphique.

6.2.1 Présentation

MySQL Control Center (MyCC) est un client graphique Open Source développé par MySQL AB. Il est encore en version de développement, mais est déjà très fonctionnel. Il est disponible pour Linux et Windows.
Plus d'informations sont disponibles sur http://www.mysql.com/products/mycc/index.html

6.2.2 Installation

Une distribution binaire existe pour linux, les étapes à suivre pour l'installer sont donc les suivantes :

1. Télécharger un fichier-archive nommé mycc-VERSION-linux.tar.gz, où VERSION correspond au numéro de version de MyCC.
2. Désarchiver et décompresser l'archive dans votre répertoire d'installation, par exemple `/usr/local` :
   ```
   # tar xvzf mycc-VERSION-linux.tar.gz
   ```
3. Lancer l'exécutable **mycc** qui se trouve dans le dossier mycc-VERSION-linux pour utiliser MyCC.

6.2.3 Connexion

Avant de pouvoir utiliser MyCC, il faut d'abord définir une connexion en allant dans le menu **file** puis **Register server**. On obtient la fenêtre de la figure 6.9.

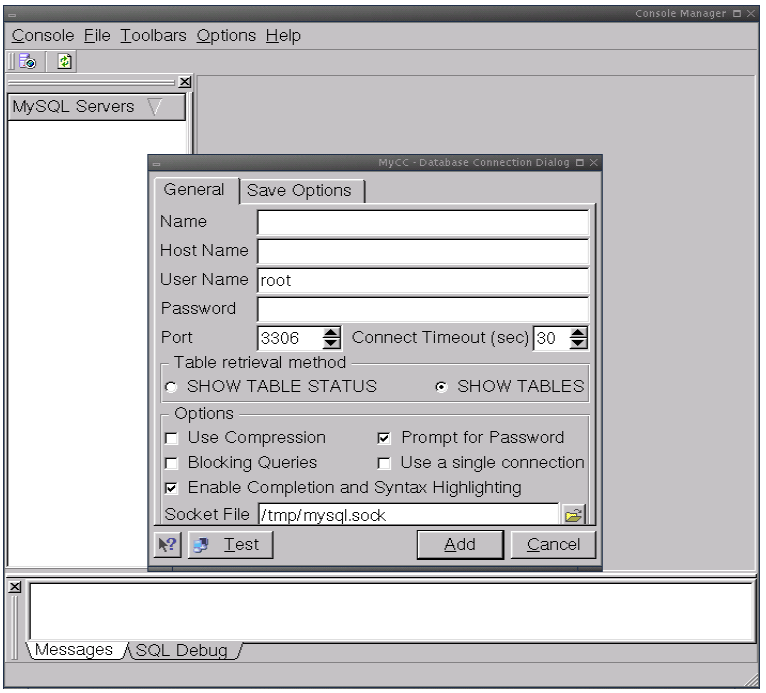


FIG. 6.9: *MyCC : définition d'un profil de connexion*

Les champs à remplir correspondent aux différentes options disponibles avec le client texte : l'adresse du serveur, l'utilisateur et mot de passe MySQL, l'utilisation ou non de la compression ... Il est possible de définir plusieurs profils de connexions différentes.

6.2.4 Administration du serveur

Ensuite pour se connecter, double-cliquer sur un des profils définis

Affichage des paramètres du serveur

L'outil graphique MyCC propose quelques fonctionnalités telles que l'affichage des variables du serveur, de l'état de la connexion, la gestion des threads... Par exemple, la commande **show variables** est disponible dans le menu contextuel de **Server Administration** comme sur la figure 6.10.

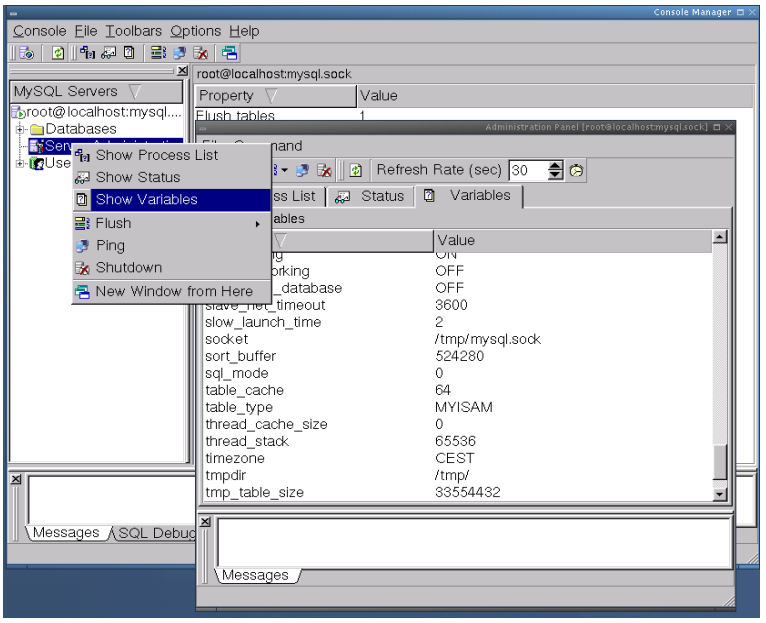


FIG. 6.10: *MyCC : affichage des variables*

Ou encore la liste des threads en cours d'exécution :

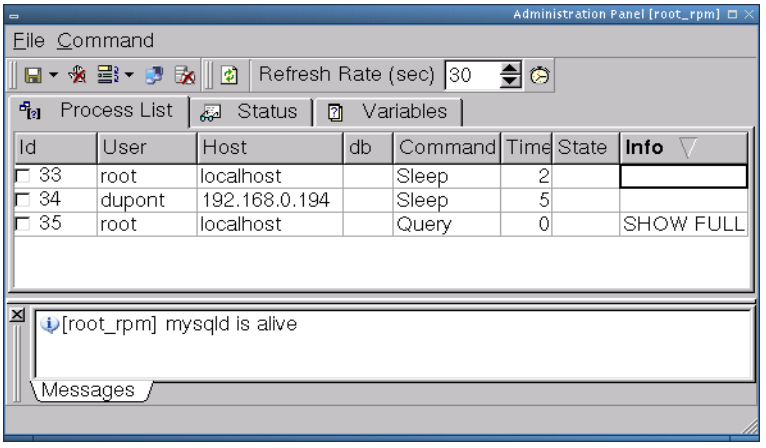


FIG. 6.11: *MyCC : affichage des threads en cours d'exécution*

A partir de cette fenêtre, il est possible de tuer un thread en le sélectionnant puis en cliquant sur l'icône **Kill Process**. On peut aussi arrêter le serveur, lui envoyer un ping pour vérifier qu'il est en cours d'exécution...

Les commandes flush

La commande **Flush** est disponible dans le menu contextuel de **Server Administration**, comme le montre la figure ci-dessous :

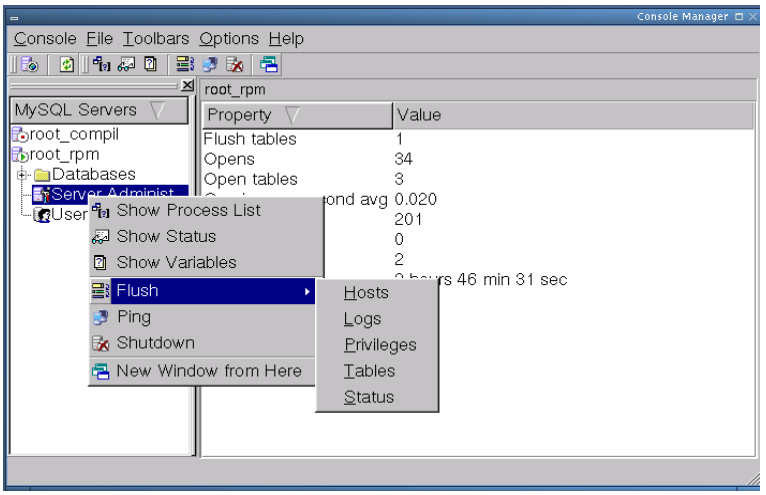


FIG. 6.12: *MyCC : comment purger le cache de MySQL*

Cette commande permet de réinitialiser différentes parties du serveur MySQL, comme par exemple les privilèges.

6.2.5 Gestion des privilèges

Le client graphique de MySQL permet de gérer les privilèges.

Dans la capture ci-dessous, on ajoute un utilisateur *dupont* n'ayant accès qu'à la base de données *linagora*, et ayant tous les droits sur cette base.
A noter que juste après l'installation de MySQL, tous les utilisateurs, depuis tous les domaines ont tous les droits sur la base *test* et les bases commançant par *test_*.

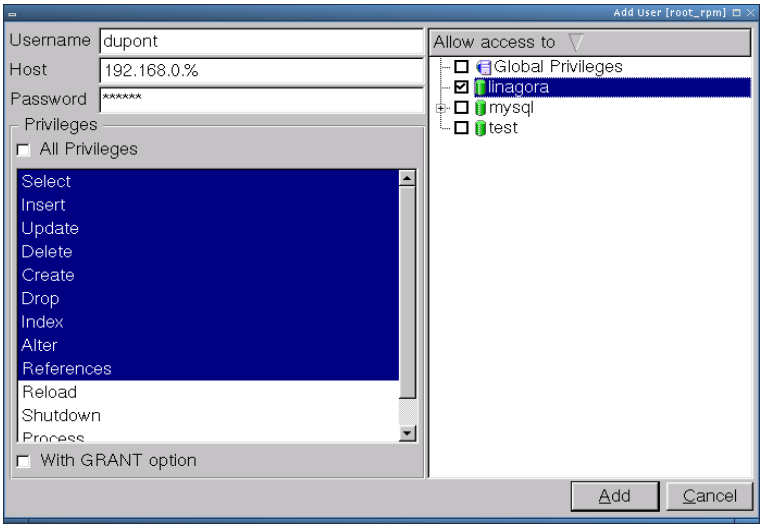


FIG. 6.13: *MyCC : ajout d'un utilisateur*

De même il est possible de supprimer des privilèges à un utilisateur, de supprimer un utilisateur, de lui ajouter des privilèges par la suite, ... de façon simple grâce à la section **User Administration** :

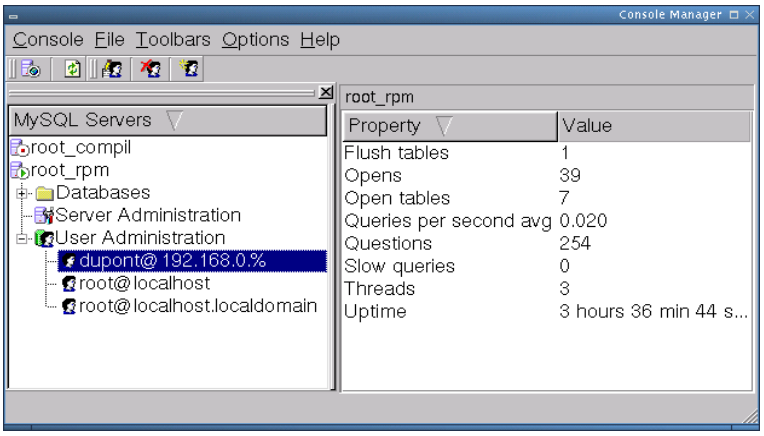


FIG. 6.14: *MyCC : administration des utilisateurs*

Atelier 6 : Outils d'administration graphiques

Exercice 6.1 *Installation de MyCC*

1. Installez MyCC et définissez un profil de connexion root.
2. Connectez-vous en root, et créez une base de données **MYCC**. Dans cette base ajoutez une table **profil**, comportant les champs suivants : IdProfil (entier) , Nom (Chaîne de caractères), Prenom (Chaîne de caractères), AdresseServeur (Chaîne de caractères).
3. Ajoutez un utilisateur **jacques**, ne pouvant se connecter que depuis la machine locale, n'ayant accès qu'à la base de données MYCC, et ayant tous les privilèges sur cette base.
4. Créez un profil de connexion pour l'utilisateur jacques, et connectez-vous au serveur MySQL en tant que jacques.