PARTIE V: INITIATION A MYSQL

MySQL est un SGBDR (Système de Gestion de Bases de Données Relationnelles)

client-serveur :

le serveur est une application installée sur l'ordinateur où sont stockées les données, il attend des requêtes des clients, accède à la base pour exécuter la requête du client et lui fournir une réponse ; le client est un programme installé sur votre ordinateur qui se connecte par l'intermédiaire d'un réseau au serveur pour effectuer une requête ;

MySQL comporte :

- un serveur SQL (Structured Query Language) ;
- des programmes clients pour accéder au serveur (mysql, mysqldump,etc.) ;
- des outils d'administration ;

Avantages

- multi-utilisateurs ;
- Gratuit,multiplateforme
- interfaces de programmation (API) : C, Perl, PHP, Python et Java ;

QUELQUES UTILITAIRES DE MYSQL

Le système de base de données de mysql vient avec les programmes suivants (mysql ,mysqlaccess ,mysqladmin ,mysqld ,mysqldump , mysqlshow)

Ces commandes ont en commun les options suivantes :

- -- help	L'information d'utilisation
- h, -- hostname=[hostname]	Le nom du serveur
- P, -- port=[port]	Le port à employer en se reliant au moteur de base de données de mysql.
- p, -- password=[password]	le mot de passe du compte d'utilisateur que vous souhaitez vous relier
- u, -- user=[user]	le nom du compte d'utilisateur
- V, -- version	version.

COMMANDE MYSQLD : Le programme de serveur de mysql.

COMMANDE MYSQLADMIN

pour administrer de divers aspects du système de base de données de mysql

shell> mysqladmin [OPTIONS] command [command-option] command ...

Quelques options supportées par mysqladmin avec la commande **mysqladmin –help.**

create databasename	Crée une nouvelle base
drop databasename	Efface une base et toutes ces tables
flush-tables	Vide de la mémoire toutes les tables
flush-privileges	:Recharger les tables de droits (identique à la commande reload).
password	Spécifie un nouveau mot de passe.
ping	Vérifie si mysqld fonctionne ou pas.
processlist	Affiche la liste des processus du serveur
reload	Recharge les tables de droits.
shutdown	: Eteind le serveur.
status	Affiche le message de statut court du serveur.

COMMANDE MYSQLDUMP

1)Pour sauvegarder un ensemble de bases de données

shell> mysqldump -h hote -u durant -p mot_de_passe base1 Base2 > save.mysql

2)Pour sauvegarder toutes les bases de données du serveur

shell> mysqldump -h hote -u durant -p --all-databases > save.mysql

3)Pour restaurer une base de données

shell> mysqldump -h hote -u durant -p base < sauve.sql

shell> mysqldump –host= hote --user= durant -p base < sauve.sql

BASE.SQL

```
-- MySQL dump 10.11
-- Host: localhost    Database: base
-- Server version   5.0.41-community-nt
--
-- Table structure for table `client`
--
DROP TABLE IF EXISTS `client`;
CREATE TABLE `client` (
  `codecli` char(3) NOT NULL,
  `nom` char(10) default NULL,
  `adresse` char(12) default NULL,
  `solde` int(11) default NULL,
  PRIMARY KEY  (`codecli`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `client`
--
LOCK TABLES `client` WRITE;
/*!40000 ALTER TABLE `client` DISABLE KEYS */;
INSERT INTO `client` VALUES ('c1','rabe','tana',100),('c2','koto','tana',100),('c3','jean',
'diego',200);
UNLOCK TABLES;
```

```
--
-- Table structure for table `commande`
--

DROP TABLE IF EXISTS `commande`;
CREATE TABLE `commande` (
  `codecli` char(5) NOT NULL,
  `codepro` char(5) NOT NULL,
  `qtecom` int(11) default NULL,
  PRIMARY KEY  (`codecli`,`codepro`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `commande`
--

LOCK TABLES `commande` WRITE;
/*!40000 ALTER TABLE `commande` DISABLE KEYS */;
INSERT INTO `commande` VALUES ('c1','p1',10),('c1','p2',10),('c1','p3',20),('c2','p1',20)
,('c2','p2',30),('c3','p3',30);
/*!40000 ALTER TABLE `commande` ENABLE KEYS */;
UNLOCK TABLES;
```

```
--
-- Table structure for table `produit`
--

DROP TABLE IF EXISTS `produit`;
CREATE TABLE `produit` (
  `codepro` char(5) NOT NULL,
  `design` char(20) default NULL,
  `pu` int(11) default NULL,
  PRIMARY KEY  (`codepro`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `produit`
--

LOCK TABLES `produit` WRITE;
/*!40000 ALTER TABLE `produit` DISABLE KEYS */;
INSERT INTO `produit` VALUES ('p1','vary',200),('p2','mofo',100),('p3','akondro',100);
/*!40000 ALTER TABLE `produit` ENABLE KEYS */;
UNLOCK TABLES;

-- Dump completed on 2014-06-02  3:50:08
```

Connexion à un serveur MySQL (depuis un terminal unix /DOS/ telnet)

Si l'administrateur vous a donné la permission de vous connecter à une base de données et de créer des tables (vous avez un login et un password) vous pouvez établir une connexion avec le serveur sur votre console en tapant :

$ MYSQL -h nomHote -u nomUtilisateur –pmot_de_passe **NOMBASE**

-h nomHote : ordinateur (nom logique ou adresse IP) qui héberge le serveur (par défaut **LOCALHOST =127.0.0.1**)

-u nomUtilisateur : votre nom d'utilisateur MySQL (non UNIX) (optionnel si c'est le même que votre login)

-pmot_de_passe : mysql vous demandera votre mot de passe

NOMBASE: optionnel

root: le superutilisateur de Mysql

$ MYSQL -h localhost –u root

Exemple d'une session

$ mysql -h lucky.luke.com -u jolly –p
Enter password : rantanplan
Welcome to MySQL ...

mysql> CREATE DATABASE   base1;
mysql> **USE base1;**
Mysql SELECT NOW();
mysql>SELECT NOW(),VERSION();
Mysql> SELECT * FROM Client;
mysql> DROP DATABASE IF EXISTS base2;
mysql > **SHOW DATABASES**;
mysql> USE base3;
mysql>quit;
$
$ mysql -h lucky.luke.com -u jolly -p base1

Dans un fichier

Vous pouvez écrire vos requêtes dans un fichier (par exemple toutes les requêtes pour créer vos tables) et faire lire le fichier et exécuter les requêtes à mysql en écrivant :

$ mysql -h lucky.luke.com -u jollyjumper -p nombase < fichier.sql

FICHIER .SQL(SCRIPT)

```
DELETE FROM CLIENT ;

INSERT INTO  Client  VALUES ("C1", 'RABE','Tana','1956-10-15',12000);

INSERT INTO  Client  VALUES ("C2", 'KOTO','Tana','1956-10-15',13000);
```

CREEZ VOS TABLES APRES OUVERTURE DE LA BASE

```
DROP TABLE IF EXISTS INDIVIDU,PRISON;

CREATE TABLE INDIVIDU (
nom VARCHAR(85) NOT NULL,
prenom VARCHAR(50),
signe_particulier VARCHAR(170),
prison INT NOT NULL);

// NOM DU PRISON
create table prison (
nom_prison VARCHAR(60) NOT NULL,
adresse VARCHAR(100),
id_prison INT NOT NULL AUTO_INCREMENT,
PRIMARY KEY(id_prison));

mysql> SHOW TABLES;
mysql> SHOW COLUMNS FROM  individu ;
mysql> DESCRIBE INDIVIDU;
```

Insérer, modifier, supprimer des lignes dans les tables

Insérer des lignes : INSERT

INSERT INTO nom_table (nom_col1, nom_col2, ...) VALUES (val_col1, val_col2, ...);

```
mysql> INSERT INTO individu (nom, prenom, signe_particulier, prison)
 VALUES ('Dalton', 'Joe', 'chef','3');
Query OK, 1 row affected (0.00 sec)
```

Modifier des lignes : UPDATE

UPDATE nom_table SET nom_col1=valeur1, nom_col2=valeur2, ... WHERE condition

```
mysql> UPDATE individu SET signe_particulier='grand et bete'
 WHERE nom='Dalton' AND prenom='Averell';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings:
```

Supprimer des lignes : DELETE

DELETE from nom_table WHERE condition

```
mysql> DELETE from individu WHERE nom='Dalton' AND
prenom='Jack';
Query OK, 1 row affected (0.00 sec)

mysql> select * from individu;
```

Effectuer une recherche

Jolly souhaite obtenir toutes les lignes de la table individu :
mysql> select * from individu;

Pour obtenir uniquement les colonnes nom, prenom et signe_particulier, il suffit de faire :
mysql> select nom, prenom, signe_particulier from individu;

Pour classer par ordre alphabétique en fonction du nom et du prénom :
mysql> select nom, prenom, signe_particulier from individu order by nom,prenom;

Pour savoir de quelle prison les hors-la-loi se sont enfuis :
mysql> select nom, prenom, nom_prison from individu, prison
where id_prison=prison order by nom, prenom;

Pour afficher uniquement les renseignements concernant les Dalton :
mysql> select nom, prenom, signe_particulier, nom_prison from individu, prison where id_prison=prison and nom='Dalton' order by nom, prenom;

Pour connaître la liste des évadés de la prison de Fort :
mysql> select nom, prenom, nom_prison, adresse from individu, prison
where individu.prison=prison.id_prison and prison.nom_prison='prison de Fort ';

❑Une référence à une table peut être aliasée en utilisant nom_de_table [AS] alias_name :

mysql> SELECT t1.name, t2.salary FROM employee t1, info t2 **WHERE t1.name = t2.name**

❑ L'alias est utilisé de la même façon que le nom du champ et peut être employé avec des clauses ORDER BY ou HAVING. Par exemple :

mysql> SELECT CONCAT(last_name,', ',first_name) AS full_name **FROM mytable ORDER BY full_name;**

mysql> *SELECT user,MAX(salary) FROM users GROUP BY user HAVING MAX(salary)>10;*

mysql> SELECT user, MAX(salary) AS max_salary FROM users GROUP BY user HAVING **max_salary**>10;

SQL LIMIT

La clause LIMIT permet de spécifier le nombre maximum de résultats que l'ont souhaite obtenir. Cette clause est souvent associé à un OFFSET, c'est-à-dire effectuer un décalage sur le jeu de résultat.

La syntaxe commune aux SGBD est la suivante :

SELECT *FROM table LIMIT 10 : permet de récupérer seulement les 10 premiers résultats d'une table

SELECT *FROM table LIMIT 10 OFFSET 5 permet de récupérer les résultats 6 à 15 (car l'OFFSET commence toujours à 0).

AVEC MySQL ,on peut écrire aussi **LIMIT [offset,] lignes** :
SELECT *FROM table LIMIT 5, 10 :retourne les enregistrements 6 à 15 d'une table..

Création d'une table et contraintes associées

Afin d'assurer l'intégrité de la base dans laquelle se trouve une table, on peut soumettre une table à un certain nombre de contraintes.
Les plus classiques sont les suivantes:

- un champ doit **toujours posséder une valeur**.
- un champ sert de **clef primaire** à la table.
- un champ sert de **clef étrangère**: il fait dans ce cas référence à la clef primaire d'une autre table.

- un champ doit être **unique** dans une table.
- un champ n'accepte que certaines valeurs.

Il existe plusieurs moteurs de stockage dans MySQL .

Les deux plus utiles sont:
  - **MyIsam:** c'est le moteur par défaut (qui ne gère pas l'intégrité référentielle).
  - **InnoDB**: c'est le moteur le plus évolué qui gère l'intégrité référentielle *(les clés étrangères)* et les transactions .

Il est possible de préciser le moteur que l'on souhaite au moment de la création de la table :

CREATE TABLE nom_table
(nom_colonne1 TYPE ATTRIBUTS,
   nom_colonne2 TYPE ATTRIBUTS,
   ...
   nom_colonneN TYPE ATTRIBUTS,
)[ENGINE=moteur];

- **AUTO_INCREMENT** : quand l'enregistrement est créé, un numéro unique est attribué ;
- **NOT NULL** : le champ doit être obligatoirement rempli ;
- **NULL** : les valeurs peuvent être omises ;
- **[CONSTRAINT [symbole_contrainte]] PRIMARY KEY (champ cle1 [, *champ cle2*, ...]) ou *nomChamp* PRIMARY KEY** : la colonne est une clé primaire ;
- **[CONSTRAINT [symbole_contrainte]] UNIQUE (*champ1*[, *champ2*, ...])** : la colonne est unique ;
- **UNSIGNED** : indique que l'entier est non négatif ;
- **ZEROFILL** : pour les types numériques entiers et réels, sert à compléter avec des 0 l'affichage en fonction du nombre maximal de chiffres affichés ;
- **BINARY** : pour les types chaînes (VAR)CHAR BINARY indique que les valeurs des colonnes sont sensibles à la casse pour les opérations de tri, comparaison ;
- **DEFAULT "valeur"** : valeur par défaut du champ.

mysql> SELECT * FROM personne
WHERE nom LIKE **BINARY** 'r%';

ne retourne rien puisque la première lettre des noms est toujours en majuscule.

Ajouter des contraintes à des tables déjà existantes

Après création de la table,on peut toujours utiliser ALTER TABLE

```
CREATE TABLE TbClient
(
  IDClient INT,
  nomClient VARCHAR(20),
  IDPays INT,
  numeroTelephoneClient VARCHAR(36)
)
```

=> contraintes de valuation obligatoire

ALTER TABLE TbClient
ALTER COLUMN nomClient VARCHAR(20) NOT NULL ;

=> contraintes de clé primaire

ALTER TABLE nom_table

ADD [CONSTRAINT [symbole_contrainte]] PRIMARY KEY (colonne_pk1 [, colonne_pk2, ...]);

ALTER TABLE TbClient
ADD CONSTRAINT PK_TbClient_IDClient PRIMARY KEY(IDClient)

Suppression de la clé primaire

ALTER TABLE nom_table DROP PRIMARY KEY

=> contraintes d’unicité

```
ALTER TABLE TbClient
ADD CONSTRAINT UQ_TbClient_nomClient UNIQUE
(nomClient),
  CONSTRAINT UQ_TbClient_numeroTelephoneClient UNIQUE
(numeroTelephoneClient)
```

=> contraintes de valeur par défaut :

```
ALTER TABLE TbClient
ADD CONSTRAINT DF_TbClient_datenaisClient
  DEFAULT ("1955-10-10") FOR datenaissClient
```

Types pour une colonne

Types numériques

Entiers:
TINYINT[(taille)] 1 octet
SMALLINT[(taille)] 2 octets ;
MEDIUMINT[(taille)]3 octets ;
INT[(taille)]4 octets ;
INTEGER[(taille)] synonyme de INT ;
BIGINT[(taille)]8 octets ;

L'attribut ZEROFILL est utilisé pour compléter avec des 0 les valeurs jusqu'à la taille maximale d'affichage.

Réels :
REAL[(taille,nb_decim)]synonyme de double :
DOUBLE[(taille,nb_decim)] nombre à virgule 8 octets ;
FLOAT[(taille,nb_decim)] nombre à virgule 4 octets;
DECIMAL(taille,nb_decim) nombre stocké comme une chaîne
NUMERIC(taille,nb_decim) synonyme de décimal.

Types chaînes

CHAR(taille) une chaîne de longueur fixe qui occupe le nombre d'octets indiqués par taille

VARCHAR(taille) une chaîne de longueur variable, un enregistrement est stocké avec le nombre d'octets nécessaire ;

TINYBLOB très petit Binary Large Object (longueur maximale autorisée : 28-1 octets) ;
BLOB un grand objet binaire (Binary Large Objetct), permet de stocker des images, textes, en fait des objets dont la taille varie pour la colonne (longueur maximale autorisée : 216-1 octets);
MEDIUMBLOB un Binary Large Object moyen (longueur maximale autorisée : 224-1 octets)
LONGBLOB un grand Binary Large Object (longueur maximale autorisée : 232-1 octets)

TINYTEXT chaîne texte très petite (comme TINYBLOB mais recherche insensible à la casse) ;

TEXT chaîne texte, identique à un blob mais la recherche dans un type TEXT n'est pas sensible à la casse ;

MEDIUMTEXT chaîne texte de taille moyenne (= MEDIUMBLOB) ;

LONGTEXT une grande chaîne texte (= LONGBLOB) ;

ENUM(value1,value2,value3,...) la valeur de la colonne doit obligatoirement être une des valeurs de l'énumération ;

Dates et temps

DATE une date (l'affichage sera au format 'YYYY-MM-DD') ;

YEAR une année (l'affichage sera au format 'YYYY') ;

TIME une heure (l'affichage sera au format 'HH:MM:SS') ;

TIMESTAMP une durée (l'affichage est au format 'YYYYMMDDHHMMSS', 'YYMMDDHHMMSS', 'YYYYMMDD', ou 'YYMMDD') ;

DATETIME une date et heure (l'affichage sera au format 'YYYY-MM-DD HH:MM:SS').

NB : MySQL n'est pas sensible à la casse pour les fonctions et les mots clés, ainsi CREATE est équivalent à create. Par contre les noms des bases, tables et colonnes sous Unix seront sensibles à la casse.

Depuis MySQL 3.23, vous pouvez créer une table à partir d'une autre, en ajoutant une commande SELECT après la commande CREATE TABLE :

Mysql>CREATE TABLE new_tbl SELECT * FROM orig_tbl;

mysql>CREATE TABLE clientele SELECT nom,solde FROM client;
mysql> SELECT * FROM clientele;

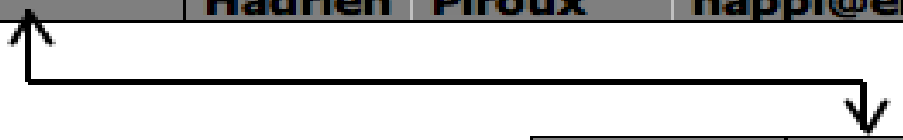
Clés étrangères

Les clés étrangères permettent la vérification de l'intégrité de la base.

Reprenons l'exemple dans lequel on a une table *Client* et une table *Commande*. Dans la table *Commande*, on a une colonne qui contient une référence au client. Ici, le client numéro 3, M. Nicolas Jacques, a donc passé une commande de trois tubes de colle, tandis que Mme Marie Malherbe (cliente numéro 2) a passé deux commandes, pour du papier et des ciseaux.

Numéro	Nom	Prénom	Email
1	Jean	Dupont	jdupont@email.com
2	Marie	Malherbe	mama@email.com
3	Nicolas	Jacques	Jacques.nicolas@email.com
4	Hadrien	Piroux	happi@email.com

Numéro	Client	Produit	Quantité
1	3	Tube de colle	3
2	2	Rame de papier A4	6
3	2	Ciseaux	2



Référence d'une table à une autre

Pour assurer l'intégrité référentielle entre 2 tables, on crée une clé étrangère. On doit choisir le moteur InnoDB qui gère l'intégrité référentielle *(les clés étrangères)*
Pour déclarer une clef étrangère, c'est-à-dire qu'un champ de votre table fait référence à un champ d'une autre table, on utilise la syntaxe:

[CONSTRAINT symbol] FOREIGN KEY (*nomChamp*)
REFERENCES *nomTable*(*nomChamp*)
[ON DELETE {CASCADE | SET NULL }]
[ON UPDATE {CASCADE | SET NULL }]

- **ON DELETE/UPDATE CASCADE** fait de manière à ce qu'une mise à jour ou une suppression dans la table parente soit impactée dans la table enfant.
- **ON DELETE/UPDATE SET NULL** fait en sorte que les données de la clé étrangère ayant perdu leur référence (suite à une modification ou une suppression) soient mises à NULL.

Exemple 1: on a 2 tables Parent et enfant en relation ONETOMANY

```
mysql> CREATE TABLE parent (
id INT NOT NULL AUTO_INCREMENT,
data varchar(20) NOT NULL, PRIMARY KEY (id)
) ENGINE = InnoDB;
```

et

```
mysql> CREATE TABLE enfant (
id int(11) NOT NULL AUTO_INCREMENT,
id_parent int(11) NOT NULL,data2 varchar(20) NOT NULL,
PRIMARY KEY (id),
CONSTRAINT fk_parent FOREIGN KEY (id_parent)
REFERENCES parent (id)
ON DELETE CASCADE ON UPDATE CASCADE)
ENGINE=InnoDB;
```

```
mysql> INSERT INTO parent (data) VALUES ('test');
mysql> INSERT INTO enfant (id_parent,data2) VALUES (1,'test');
mysql>INSERT INTO enfant (id_parent,data2) VALUES (1,'test2');

mysql> SELECT * FROM enfant;
+----+-----------+-------+
| id | id_parent | data2 |
+----+-----------+-------+
|  1 |         1 | test  |
|  2 |         1 | test2 |

mysql> UPDATE parent SET id = id +1;
mysql> SELECT * FROM enfant;
+----+-----------+-------+
| id | id_parent | data2 |
+----+-----------+-------+
|  1 |         2 | test  |
|  2 |         2 | test2 |
```

NB : Si ON DELETE SET NULL est spécifiée, la ligne parente est supprimée et la colonne de la clé étrangère des lignes filles prenne automatiquement la valeur de NULL mais assurez vous que vous **n'avez pas déclaré les colonnes de la table fille** NOT NULL

Exemple 2 :

Imaginons dans un MCD deux entités (familles, professions) et une association (prof_fam) qui va contenir les clés primaires des entités familles et professions..

Code sql :

```
CREATE TABLE familles (
nom_fam VARCHAR(3) NOT NULL PRIMARY KEY,
designation VARCHAR(30) NULL,
code VARCHAR(3) NULL
) TYPE = InnoDB;

CREATE TABLE professions (
nom_prof VARCHAR(10) NOT NULL PRIMARY KEY
) TYPE = InnoDB;
```

CREATE TABLE **prof_fam** (
nom_fam VARCHAR(3) NOT NULL,
nom_prof VARCHAR(10) NOT NULL,
PRIMARY KEY(nom_fam,nom_prof),
FOREIGN KEY(nom_fam) REFERENCES familles(nom_fam) ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY(nom_prof) REFERENCES professions(nom_prof) ON DELETE CASCADE ON UPDATE CASCADE,
INDEX(nom_fam),
INDEX(nom_prof)
) TYPE = InnoDB;

Attention l'ordre de création est important, d'abord les entités qui ne comporte pas de clés étrangères, ensuite les associations.

Ajouter et modifier des contraintes des clés étrangères

Voici comment on ajoute une clé étrangère à une table déjà existante :

```
ALTER TABLE nom_table
ADD [CONSTRAINT fk_col_ref]
FOREIGN KEY (colonne)                REFERENCES
table_ref(col_ref)
{ON UPDATE |DELETE} {SET NULL | CASCADE};
```

Donc si on imagine les tables *Client* et *Commande*, pour créer la table *Commande* avec une clé étrangère ayant pour référence la colonne *numero* de la table *Client*, on utilisera :

```
CREATE TABLE Commande (
numero INT UNSIGNED PRIMARY KEY
AUTO_INCREMENT,
client INT UNSIGNED NOT NULL,
produit VARCHAR(40),    quantite SMALLINT
DEFAULT 1
)
ENGINE=InnoDB;
```

Après création de la table

il faut utiliser ALTER TABLE

.

ALTER TABLE Commande ADD CONSTRAINT fk_client_numero FOREIGN KEY (client) REFERENCES Client(numero) ***ON DELETE CASCADE ON UPDATE CASCADE*****;**

Suppression d'une clé étrangère

Il peut y avoir plusieurs clés étrangères par table. Par conséquent, lors d'une suppression il faut identifier la clé à détruire. Cela se fait grâce au symbole de la contrainte.
ALTER TABLE nom_table DROP FOREIGN KEY symbole_contrainte
mysql> ALTER TABLE **Commande** DROP FOREIGN **fk_client_numero**

Limiter les valeurs à un ensemble prédéfini

L'option **CHECK** permet de limiter le contenu d'un champ à une liste de valeurs prédéfinies.

```
CREATE TABLE batiment (id INTEGER NOT NULL
AUTO_INCREMENT, nom VARCHAR (100) NOT NULL, annee
INTEGER CHECK (annee BETWEEN 1900 AND 1999) DEFAULT
1900, adresse VARCHAR (100) CHECK (adresse IN ('rue
Vandrezanne', 'boulevard Auriol', 'rue de Tolbiac')) NOT NULL,
id_Architecte INTEGER NOT NULL, PRIMARY KEY (id),
  FOREIGN KEY (id_architecte) REFERENCES architecte(id)
     ON DELETE SET NULL);
```

Le champ annee ne peut alors comprendre que des valeurs comprises entre 1900 et 1999.
Et le champ adresse ne peut alors comprendre que une des 3 adresses mentionnées.

Attention: la contrainte CHECK est accepté par MySQL mais ne donne lieu à aucun contrôle.
MySQL dispose de types qui permettent le contrôle: ENUM et SET (qui ne font pas partie de la norme SQL)
.

Exemple:

adresse **ENUM** ('rue Vandrezanne', 'boulevard Auriol', 'rue de Tolbiac')

Si la valeur insérée dans le champ adresse n'appartient pas à l'ensemble défini par ENUM, MySQL affecte une chaîne vide NULL.

LES INDEX

Un index est une structure qui reprend la liste ordonnée des valeurs auxquelles il se rapporte. Les index sont utilisés pour accélérer les requêtes utilisant des colonnes indexées comme critères de recherche (notamment les requêtes impliquant plusieurs tables, ou les requêtes de recherche). Les types d'index proposés par MySQL sont :

- **Key ou Index** : index simple, autorisant Null et doublons.
- **Unique** : index interdisant les doublons, et mettant ainsi en oeuvre une contrainte d'unicité.
- **Primary Key** : MySQL définit automatiquement un index sur chaque clé primaire ;

Création et suppression des index

Les index sont représentés par le mot-clé INDEX ou KEY et peuvent être créés de deux manières :

- soit directement lors de la création de la table ;
- soit en les ajoutant par la suite.

Ajout des index lors de la création de la table

deux possibilités : vous pouvez préciser dans la description de la colonne qu'il s'agit d'un index, ou lister les index par la suite.

❑ ***Index simple dans la description de la colonne en précisant si la colonne est un index.***

```
CREATE TABLE nom_table (
colonne1  KEY,        -- Crée un index simple sur colonne1
colonne2 (40) UNIQUE,-- Crée un index unique sur colonne2);
```

❑Liste d'index

L'autre possibilité est d'ajouter les index à la suite des colonnes, en séparant chaque élément par une virgule :

```
CREATE TABLE nom_table (
    colonne1 description_colonne1,
    [colonne2 description_colonne2,
    colonne3 description_colonne3,
    ...,]
    [PRIMARY KEY (colonne_clé_primaire)],
    [INDEX [nom_index] (colonne1_index [, colonne2_index, ...]]
)
[ENGINE=moteur];
```

Exemple : si l'on avait voulu créer la table *Animal* avec un index sur la date de naissance, et un autre sur les 10 premières lettres du nom, on aurait pu utiliser la commande suivante :

CREATE TABLE Animal **(** id UNSIGNED NOT NULL AUTO_INCREMENT, espece (40) NOT NULL**,** sexe **(1),** date_naissance DATETIME NOT NULL**,** nom

(30), commentaires **,** **PRIMARY KEY (**id**),**

INDEX ind_date_naissance **(**date_naissance**), -- index sur la date de naissance**

INDEX ind_nom **(**nom**(10))** **-- index sur le nom**

ENGINE=INNODB**;**

<u>Suppression d'un index</u>

ALTER TABLE nom_table **DROP INDEX** nom_index**;**

CREATION DE VUES

CREATE [OR REPLACE] VIEW *view_name* [(*column_list*)]
AS *select_statement*

Cette commande crée une nouvelle vue, ou remplace une vue existante si la clause OR REPLACE est fournie. La clause *select_statement* est une commande SELECT qui fournit la définition de la vue et peut contenir une jointure. La liste optionnelle de colonnes peut être fournie pour définir explicitement les noms des colonnes

Les exemples suivants définissent une vue qui sélectionne 2 colonnes dans une table, et leur applique une transformation :

```
mysql> CREATE TABLE t (qty INT, price INT);
mysql> INSERT INTO t VALUES(3, 50);
mysql> CREATE VIEW v AS SELECT qty, price, qty*price AS value FROM
t;
mysql> SELECT * FROM v;
| qty  | price | value |
|    3 |    50 |   150 |
```

Par défaut, la vue est placée dans la base de données par défaut. Pour créer une vue explicitement dans une base de données, spécifiez le nom de la base de données lors de la création : ***db_name.view_name***.

```
mysql> CREATE VIEW test.v AS SELECT * FROM t;
```

Cette commande a été ajoutée en MySQL 5.0.1

ALTER VIEW *view_name* [(*column_list*)] AS *select_statement :*
Cette commande modifie la définition d'une vue.
select_statement est le même que pour CREATE VIEW.

DROP VIEW [IF EXISTS] *view_name* [, *view_name*] ...

DROP VIEW supprime une ou plusieurs vues. Vous devez avoir les droits de DROP pour chaque vue.

Vous pouvez utiliser le mot clé IF EXISTS pour éviter l'affichage d'un message d'alerte lorsque les vues n'existent pas.

Informations sur les tables

Liste des tables existantes :
SHOW TABLES;

Liste des colonnes avec leurs types, les trois instructions ci-dessous sont équivalentes :

DESCRIBE nomtable;

EXPLAIN nomtable;

SHOW COLUMNS FROM nomtable;

Suppression d'une table

DROP TABLE [IF EXISTS] nom_table1, nom_table2, ...

Modification d'une table

ALTER TABLE nom_table alteration

ALTER permet de changer le nom de la table, de modifier les types des colonnes, et les noms des colonnes, d'ajouter des colonnes, d'en supprimer. alteration peut prendre les valeurs : ADD, ALTER, CHANGE, MODIFY, DROP, RENAME.

Modifier le type d'une colonne

MODIFY nom_colonne definition_colonne

Nous souhaitons modifier le type de la colonne nom en char :
mysql> alter table individu modify nom char;

mysql> describe individu;

Modifier le nom et le type de la colonne

CHANGE ancien_nom nouveau_nom definition_col

Par exemple ou peut décider de rebaptiser la colonne nom en nom_hors_la_loi et de lui attribuer le type varchar(86)

mysql> **alter table individu change nom nom_hors_la_loi varchar(86);**

Query OK, 7 rows affected (0.01 sec)
Records: 7 Duplicates: 0 Warnings: 0
mysql> describe individu;

Ajouter une colonne

ADD nom_colonne definition_colonne [FIRST | AFTER nom_colonne]

Si on souhaite ajouter en début de la table une colonne pour stocker l'age :

mysql> **alter table individu add age TINYINT UNSIGNED FIRST;**

Query OK, 7 rows affected (0.01 sec)
Records: 7 Duplicates: 0 Warnings: 0
mysql> describe individu;

mysql> **alter table individu add age TINYINT UNSIGNED AFTER prenom;**

Supprimer une colonne

DROP nom_colonne
Pour supprimer la colonne age :

mysql> alter table individu drop age;
mysql> describe individu;

Changer le nom de la table

RENAME AS nouveau_nom

Par exemple, pour renommer la table individu en hors-la-loi on écrira :

mysql> ALTER TABLE individu RENAME as horslaloi;
mysql> show tables;

EXPRESSIONS

Opérateurs de Comparaison

Les opérateurs relationnels: = , != différent ,
<> différent ;< ;> ;<= ; >=

- expr IN (val1, val2, ...) l'expression est une des valeurs données
- expr BETWEEN inf AND sup l'expression est entre inf et sup
- IS [NOT] NULL vrai si l'expression a la valeur NULL
- "chaine" [NOT] LIKE "exp" vrai si la chaine entière correspond à l'expression rationnelle

"toto" LIKE "to" retournera la valeur 0
"toto" LIKE "to%" retournera la valeur 1.

FONCTIONS TEMPS	EXEMPLE
•* **CURDATE(), CURRENT_DATE** Retourne la date courante au format 'YYYY-MM-DD' ou YYYYMMDD, suivant le contexte numérique ou chaîne : •* **NOW()** Retourne la date courante au format 'YYYY-MM-DD HH:MM:SS' ou YYYYMMDDHHMMSS, suivant le contexte numérique ou chaîne :	mysql> **SELECT CURDATE();** -> '1997-12-15' mysql> **SELECT NOW();** -> '1997-12-15 23:50:26'
•**CURTIME()** Retourne l'heure courante au format 'HH:MM:SS' or HHMMSS suivant le contexte numérique ou chaîne :	mysql> **SELECT CURTIME();** -> '23:50:26'

•DATEDIFF(expr,expr2) retourne le nombre de jours entre la date de début expr et la date de fin expr2. expr et expr2 sont des expressions de type DATE ou DATETIME.	mysql> **SELECT DATEDIFF('1997-12-31 23:59:59','1997-12-30');** -> 1
•DATE_FORMAT(date,format) Formate la date date avec le format format.	Par exemple, Mysql> **SELECT date_format(" 2001-01-10","%e %M %Y")** donnera 10 January 2001 mysql> **SELECT DATE_FORMAT('1997-10-04 22:23:00', '%W %M %Y');** -> 'Saturday October 1997' mysql> **SELECT DATE_FORMAT('1997-10-04 22:23:00', '%H:%i:%s');** -> '22:23:00'

•**DAYNAME(date)** Retourne le nom du jour de la semaine de date :	mysql> **SELECT DAYNAME('1998-02-05');** -> 'Thursday'
•**DAYOFMONTH(date)** Retourne le jour de la date date, dans un intervalle de 1 à 31 :	mysql> **SELECT DAYOFMONTH('1998-02-03');** -> 3
•**DAYOFWEEK(date)** Retourne l'index du jour de la semaine : pour date (1 = Dimanche, 2 = Lundi, ... 7 = Samedi).	mysql> **SELECT DAYOFWEEK('1998-02-03');** -> 3
DAYOFYEAR(date) Retourne le jour de la date date, dans un intervalle de 1 à 366 :	mysql> **SELECT DAYOFYEAR('1998-02-03');** -> 34

•**MONTH(date)** Retourne le numéro du mois de la date date, dans un intervalle de 1 à 12 :	mysql> **SELECT MONTH('1998-02-03');** -> 2
•**MONTHNAME(date)** Retourne le nom du mois de la date date :	mysql> **SELECT MONTHNAME("1998-02-05");** -> 'February'
YEAR(date) retourne la partie année de la date au format YYYY-MM-DD, dans un intervalle de 1000 à 9999:	mysql> **SELECT YEAR('2000-01-01')** -> 2000, mysql> **SELECT YEAR('98-02-03');** -> 1998
WEEk(date) retourne la semaine de l'année pour la date donnée	mysql> **SELECT WEEK('2000-01-01',2);** -> 52

FONCTIONS CHAINES	
•**lcase(ch) et lower(ch)** conversion en minuscules •**ucase(ch), upper(ch)** conversion en majuscules	mysql> **SELECT UPPER('Hey');** 'HEY' mysql> **SELECT UCase(nom) AS NomMaj FROm client;**

COMPARAISONS

•IF(expr1, expr2, expr3) si expr1 alors retourne expr2 sinon retourne expr3

```
mysql> SELECT IF(1>2,2,3);
    -> 3
mysql> SELECT IF(1<2,'oui','non');
    -> 'oui‘
mysql> SELECT nom,moy, IF(moy>=10,
‘admis',‘echec') AS Resultat from
etudiant;
```

Nom	moy	resultat
Koto	12	admis
Rivo	9	Echec

•IFNULL(expr1,expr2)
Si l'argument expr1 n'est pas NULL, la fonction IFNULL() retournera l'argument expr1, sinon elle retournera l'argument expr2.

```
mysql> SELECT IFNULL(1,0);
    -> 1
mysql> SELECT IFNULL(NULL,10);
    -> 10
 mysql> SELECT IFNULL(1/0,'oui');
    -> 'oui‘
mysql> SELECT nom, IFNULL(moy,‘Note
nulle') AS Resultat from etudiant;
```

LES EXPRESSIONS REGULIERES

MySQL fournit l'opérateur **REGEXEP (motif)** qui permet d'utiliser les expressions régulières .Une expression régulière (regex) est la meilleure méthode pour spécifier une recherche **complexe.** Le principal avantage est de pouvoir construire des requêtes beaucoup plus complexes qu'un simple LIKE en exploitant toute la puissance des expressions régulières.

Attention: REGEX n'est pas un standard SQL.

Quelques exemples basiques

SELECT * FROM clients WHERE nom LIKE 't%' ;
SELECT * FROM clients WHERE nom REGEXP '^t' ;

SELECT * FROM clients WHERE nom LIKE '%d' ;
SELECT * FROM clients WHERE nom REGEXP "d$";

SELECT * FROM clients WHERE nom LIKE '%Ram%;
recherche toutes les personnes dont le nom contient Ram
SELECT * FROM clients WHERE nom REGEXP 'Ram'

Exemples plus difficile à réaliser avec un simple LIKE :

SELECT nom,prenom from table WHERE prenom **REGEXP '(Alain|Pierre|Jean)';** la regexp (Alain|Pierre|Jean) permet de récupérer tous les noms des client dont le prénom contient Alain ou, Pierre ou Jean.
on pourrait écrire les équivalents sans REGEXP :
SELECT nom,prenom FROM client WHERE prenom='Alain' OR prenom='Jean' OR prenom='Pierre';
SELECT dep,nom,prenom FROM client WHERE prenom IN('Alain','Pierre','Jean')"

D'autres exemples,
SELECT * FROM clients WHERE nom **REGEXP '^.{4}$'** ;
recherche les clients dont le nom contient 4 caractères
SELECT code_article from produit where code_article **REGEXP '([2-9]{4})';** récupére tous les codes qui contiennent 4 chiffres mais uniquement de 2 à 9

Voici un exemple encore plus complexe : la regexp **B[an]*s** trouve l'une des chaînes suivantes Bananas , Baaaaas , Bs , et n'importe quelle autre chaîne commençant par un B , se terminant par un s , et contenant n'importe quel nombre de a et de n au milieu.

Une expression régulière peut utiliser l'un des caractères spéciaux ou constructions suivants :

▪**^ :**Correspond au début de la chaîne.

```
mysql> SELECT "fo\nfo" REGEXP  BINARY "^fo$";  -> 0
mysql> SELECT "fofo" REGEXP "^fo";              -> 1
```

▪**$:**Correspond à la fin de la chaîne.

```
mysql> SELECT "fo\no" REGEXP "^fo\no$";        -> 1
mysql> SELECT "fo\no" REGEXP "^fo$";           -> 0
```

- **. (point) :N'importe quel caractère (nouvelle ligne inclus).**

Par exemple,

"^.{3}$": chaîne qui contient 3 caractères

a.{2,2}z => abcz, aXXz, ak0z, ...

123.5 => 123.5, 12345, 123s5, 123-5

mysql> SELECT "fofo" REGEXP "^f.*"; -> 1
mysql> SELECT "fo\nfo" REGEXP "^f.*"; -> 1

a* : Correspond à toute séquence de zéro ou plusieurs caractères a .

mysql> SELECT "Ban" REGEXP "^Ba*n"; -> 1
mysql> SELECT "Baaan" REGEXP "^Ba*n"; -> 1
mysql> SELECT "Bn" REGEXP "^Ba*n"; -> 1

- **a+ :**Correspond à toute séquence de un ou plus caractères a .

mysql> SELECT "Ban" REGEXP "^Ba+n"; -> 1

mysql> SELECT "Bn" REGEXP "^Ba+n"; -> 0

a? : Correspond à zéro ou un caractère a .

mysql> SELECT "Bn" REGEXP "^Ba?n"; -> 1

mysql> SELECT "Ban" REGEXP "^Ba?n"; -> 1

mysql> SELECT "Baan" REGEXP "^Ba?n"; -> 0

- **de|abc :** Correspond aux séquences de de ou de abc

```
mysql> SELECT "pi" REGEXP "pi|apa";          -> 1
mysql> SELECT "axe" REGEXP "pi|apa";          -> 0
mysql> SELECT "apa" REGEXP "pi|apa";          -> 1mysql>
SELECT "apa" REGEXP "^(pi|apa)$";       -> 1
mysql> SELECT "pi" REGEXP "^(pi|apa)$";       -> 1mysql>
SELECT "pix" REGEXP "^(pi|apa)$";       -> 0
```

(abc)* : Correspond à zéro ou plus séquences de abc .

```
mysql> SELECT "pi" REGEXP "^(pi)*$";          -> 1
mysql> SELECT "pip" REGEXP "^(pi)*$";         -> 0
mysql> SELECT "pipi" REGEXP "^(pi)*$";        -> 1
```

▪**Les accolades {X,Y}** permettent de donner des limites de nombre.

"abc{2}": chaîne qui contient "ab" suivie de deux "c" ("abcc")
"abc{2,}": chaîne qui contient "ab" suivie de deux "c" ou plus ("abcc" etc..)
"abc{2,4}": chaîne qui contient "ab" suivie 2, 3 ou 4 "c" (" abcc" .. "abcccc")

abc{2,4}d => abccd, abcccd, abccccd et rien d'autre

a* Peut être écrit a{0,} .
a+ Peut être écrit a{1,} .
a? :Peut être écrit a{0,1}

▪ **Les crochets [ ]** définissent une liste de caractères autorisés (ou interdits). Le signe **-** permet quand à lui de définir un intervalle. Le caractère **^** après le premier crochet indique quand à lui une interdiction.

"[abc]": chaîne qui contient un "a", un "b", ou un "c"
"[a-z]": chaîne qui contient un caractère compris entre "a" et " z"

"^[a-zA-Z0-9]": chaîne qui commence par une lettre ou un chiffre
"^[^a-zA-Z]": chaîne qui ne commence pas par une lettre
a[a-d]z => aaz, abz, acz, adz et rien d'autre
exemple[4-8] => exemple4, exemple5, exemple6, exemple7, exemple8 et rien d'autre
[a-dX] , [^a-dX] : trouve n'importe quel caractère qui est (ou n'est pas, si ^ est utilisé) a , b , c , d ou X .

Dans les crochets, chaque caractère représente ce qu'il est. Pour inclure le caractère litéral] , il doit suivre immédiatement le crochet ouvrant **[** . Pour inclure le caractère litéral **-** , il doit être écrit en premier ou en dernier.

Par exemple**,**
"[\+?{}.]" : chaîne qui contient un de ces six caractères«

[ ]-]": chaîne qui contient le caractère "]" ou le caractère "-"

[a-z?+*{}]: on peut mettre une lettre, un point d'interrogation, un signe +, etc.

[a-z0-9-] permet de chercher une lettre, un chiffre ou un tiret.

```
mysql> SELECT "aXbc" REGEXP "[a-dXYZ]";        -> 1

mysql> SELECT "aXbc" REGEXP "^[a-dXYZ]$";      -> 0

mysql> SELECT "aXbc" REGEXP "^[a-dXYZ]+$";     -> 1

mysql> SELECT "aXbc" REGEXP "^[^a-dXYZ]+$";    -> 0

mysql> SELECT "gheis" REGEXP "^[^a-dXYZ]+$";   -> 1

mysql> SELECT "gheisa" REGEXP "^[^a-dXYZ]+$";  -> 0
```

II. SYSTEME DE PRIVILEGES (DROITS) DE MYSQL

•L'administration du serveur de Mysql implique l'entretien du serveur de base de données (hôtes ,utilisateurs et base de données).

•La fonction primaire du système des droits est d'authentifier un utilisateur se connectant au serveur Mysql (en indiquant l'hôte qui abrite le serveur , le nom d'user et le mot de passe) et l'associer avec les droits d'utilisation.

•Les noms d'utilisateurs et mots de passe utilisés par Mysql pour l'authetification , n'ont rien à voir avec ceux d'Unix et Windows.

•Pour conserver les informations de connexion et les droits , le serveur les tables **USER, DB et HOS**T de la base MYSQL.

•La table **USER** détermine le droit de connexion.

•Les tables **DB et HOST** sont utilisées ensembles :

- La table **DB** détermine quelles bases sont accessibles à quels utilisteurs. Les champs de droits déterminent quels sont les droits autorisés.

- La table **HOST** est utilisée comme une extension de DB lorsque vous voulez qu'une ligne de DB s'applique à plusieurs hôtes. Par exemple, si vous voulez qu'un utilisateur soit capable d'accéder à la base depuis plusieurs hôtes différents, laissez le champ Host de la table DB vide, puis ajoutez un enregistrement dans la table HOST pour chaque hôte à autoriser.

•Structure de la table USER

Host char (60)	Nom de la machine depuis laquelle est fait l'appel
User char (16)	Login (nom) de l'utilasateur
Password char (16)	Mot de passe codé
Select_priv	Droits d'effectuer des requêtes sélect (valeur Yes ou Non)
Insert_priv	Droit d'insertion d'enregistrement
Update_priv	Droit de modification
Delete_priv	Droit de suppression
Create_priv	Droits d'effectuer une création de table
Drop_priv	Droits de suppression d'une table
Reload_priv	Droits de relancer le serveur MySQL
Shutdown_priv	Droits d'arrêter le serveur MySQL
Process_priv	Droit de voir les commandes des autres users sans cryptage , notamment les commandes de changement de mot de passe
File_priv	Droit de lire ety écrire des fichiers sur le serveur en utilisant les commandes LOAD DATA INFILE et SELECT .. INTO OUTFILE.
Grant_priv	Droit de donner des droits que vous possédez à un autre user
Index_priv	Droit d'indexer une table
Alter_priv	Droit d'utiliser la commande ALTER TABLE

•Cette structure peut être consultée très simplement par **mysql> desc user ou mysql>select * from user ;**

•**Structure de la table User**

•**User(Host,User,Password,Select_priv,Insert_priv,Update_priv,Delete_Priv, Create_Priv,Drop_Priv, Reload_priv,Shutdown_priv, Process_priv, File_priv, Grant_priv, Index_priv , Alter_priv)**

•Host peut être un nom d'hôte, ou une adresse IP ou 'localhost* pour indiquer la machine locale.

•Vous pouvez utilisez les caractères spéciaux " %" et "_ " dans le champs Host.

•Host qui vaut % accepte tous les hôtes de se connecter avec votre serveur . Host vide est équivalent à '%'.

•Les caractères spéciaux ne sont pas autorisés dans le champs User, mais vous pouvez le laisser vide, ce qui équivaudra à ' %'.

•

•**Structure de la table DB (Host,Db,User,Select_priv,Insert_priv, Update_priv,Delete_Priv,Create_Priv,Drop_Priv…)**

•**Structure de la table HOST(Host,Db,Select_priv,Insert_priv,Update_priv, Delete_Priv,Create_Priv,Drop_Priv)**

•Les tables DB et HOST donnent des droits spécifiques aux bases de données. Les valeurs acceptées dans les champs sont les suivantes :

•Les caractères spéciaux " %" et "_ "peuvent être utilisés dans les champs Host et Db de deux tables.

•Un '%' dans le champ Host de la table db signifie "tous les hôtes.

•Un '%' ou une chaîne vide dans le champs Host de la table host signifie" tous les hôtes."

•Un '%' ou une chaîne vide dans le **champ Db** dans l'une des tables signifie " toutes les bases de données.

```
mysql> INSERT INTO HOST
Host,Db,User,Select_priv,Insert_priv,Update_priv,
Delete_priv,  Create_priv,Drop_priv)
    -> VALUES
 -> (‘machine1.gov','expenses','custom','Y','Y','Y','Y', 'Y','Y');

mysql> INSERT INTO  HOST
(Host,Db,User,Select_priv,Insert_priv,Update_priv,
Delete_priv, Create_priv,Drop_priv)
    -> VALUES
    -> (‘machine2.gov',expenses','custom','Y','Y','Y','Y', 'Y','Y');
```

•**Affectation de mot de passe**

Etant donné que l'installation initiale est très ouverte, la première chose à faire est d'attribuer un mot de passe au root. Vous pouvez le faire simplement avec la commande suivante :

Shell>mysql-u root mysql
Mysql>update user SET **password=PASSWORD (nouveau_mot_de_passe**) WHERE user='root';
Mysql>FLUSH PRIVILEGES;

•Un autre moyen d'attribuer le mot de passe est de passer la commande mysqladmin:

Shell> **mysqladmin-u root password Nouveau_mot_de_passe**

Ajouter de nouveaux utilisateurs à MySQL

Vous pouvez ajouter des utilisateurs de deux manières différentes : en utilisant la commande GRANT (la meilleure) ou en manipulant directement les tables de droits MyQSL.

Il y a également beaucoup de programmes contribués comme **phpmyadmin** pouvant être employé pour créer et administrer des utilisateurs

Exemple :

Les exemples suivants montrent comment employer mysql client pour installer de nouveaux utilisateurs.
Vous devez vous connecter en tant que root

```
shell> mysql --user=root mysql
mysql> GRANT ALL PRIVILEGES ON *.* TO monty@localhost
    ->     IDENTIFIED BY 'some_pass' WITH GRANT OPTION;

mysql> GRANT ALL PRIVILEGES ON *.* TO monty@"%"
    ->     IDENTIFIED BY 'some_pass' WITH GRANT OPTION;

mysql> GRANT SELECT, INSERT,UPDATE ON Base.* TO  rivo@localhost
->    IDENTIFIED BY ‘rivo’  ,  randria@localhost  IDENTIFIED BY ‘randria’ ;

mysql> GRANT SELECT, INSERT ON Base.Client  TO  koto@’192.1.1.7’
->    IDENTIFIED BY ‘koto’ ;
```

Vous pouvez également ajouter la même information d'accès d'utilisateur directement par la commande INSERT et puis en forçant le serveur à recharger ces tables.

```
shell> mysql --user=root mysql
mysql> INSERT INTO user VALUES('localhost','monty',
PASSWORD('some_pass'),
    ->          'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');

mysql> INSERT INTO user VALUES('%','monty',
PASSWORD('some_pass'),
    ->          'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');

mysql> INSERT INTO user SET Host='localhost',User='rivo',
    ->           Select_priv='Y', Insert_priv='Y';

mysql> FLUSH PRIVILEGES;
```

GRANT et REVOKE

Ajout des utilisateurs avec les privilèges

GRANT priv_type [(column_list)] [, priv_type [(column_list)] ...] ON {Nom_table | * | *.* | Nom_bdd.*} TO user_name [IDENTIFIED BY 'password'] [, user_name [IDENTIFIED BY 'password'] ...] [WITH GRANT OPTION]

Retrait des privilèges

REVOKE priv_type [(column_list)] [, priv_type [(column_list)] ...] ON {Nom_table | * | *.* | Nom_bdd.*} FROM user_name [, user_name ...]

Exemple : supprimer à l'utilisateur rivo le droit de mise à jour et à koto le droit d'insertion sur la base BASE.

mysql> ***REVOKE UPDATE ON BASE.* FROM rivo@localhost;***

mysql> ***REVOKE INSERT ON BASE.Client FROM koto@'192.1.1.7';***

Devoir SQL

Base de données

- Client (CODECLI, NOM,VILLE, DATENAIS, SOLDE)
- Produit (CODEPRO, LIBELLE,PU)
- Commande (CODECLI , CODEPRO, QTE)

Ecrire des requêtes SQL de type SELECT permettant de:

- Lister les clients de Tana par ordre décroissant de solde

- Lister les clients nés en 1954

- Lister les codes et noms des clients de Tana ayant le solde minimal

- Lister les produits (libelle,qte,pu,montant=qte*pu) commandés par le client « rabe »

- Lister les clients (nom,libelle,qte) ayant commandé du riz dont la quantité >100
- Calculer le nombre et la quantité totale des produits commandés par le client « rabe »
- Calculer le nombre et quantité totale des produits commandés par les clients de Tana
- Calculer le nombre et quantité totale des produits commandés par chaque client de Tana
- Calculer le montant total des produits commandés par chaque client (nom, montant total)
- Lister les clients dont le montant total des commandes est supérieur à 10000

EXERCICE II (MYSQL)

I Soient les tables suivantes :

Table ETUDIANT

CODET	NOM	CLASSE	BOURSE	Datenais
E1	RABE	Premiere	1000	1956-10-15
E2	RANDRIA	Seconde	2000	1975-10-12
E3	KOTO	Premiere	1500	1975-11-2
E4	RIVO	Terminale	3000	1958-10-13
E5	NONO	Terminale	1000	1956-10-15

Table MATIERE

CODEMAT	LIBELLE	COEF
M1	Maths	5
M2	Physique	3
M3	Anglais	2
M4	BD	2

Table NOTES

CODET	CODEMAT	Note
E1	M1	15
E1	M2	10
E1	M3	12
E2	M1	08
E2	M2	10
E2	M3	05
E3	M1	09
E3	M2	11
E4	M1	18

On demande de formuler des requêtes SQL pour répondre aux questions suivantes :

1)Modifier la bourse de "RABE" par 10000 F et sa classe par " Seconde«

2) Doubler toutes le bourses

4)Supprimer la matière BD

5)visualiser les étudiants de classe Seconde

5) Convertir en majuscule les noms des étudiants

7)afficher les étudiants ayant une bourse supérieure à 2000

8) afficher les 3 premiers étudiants

9) afficher les étudiants nés en 1975

10) afficher les étudiants nés au mois d'Octobre

11) Doubler la note des étudiants en maths inférieure à 5

12) afficher la note min et max en maths

12) afficher les noms des étudiants ayant la note max en maths

13) lister les matières , notes et notes pondérées obtenues par l'étudiant "RABE"

14) lister par ordre alphabétique les étudiants ayant obtenu une note >=10 en maths

15) calculer le nombre d'étudiants, le total des bourses par classe

16) calculer la moyenne générale avec les observations (ADMIS, ECHEC) obtenues par chaque étudiant

17) afficher les étudiants ayant obtenu la moyenne >=10

18) Afficher le nombre des étudiants admis

19) Afficher le premier étudiant avec la moyenne

20) Afficher le dernier étudiant avec la moyenne
21) Afficher les noms des étudiants ayant plus de 2 notes
22) Afficher les noms des étudiants n'ayant pas obtenu de notes
23) Afficher les noms des étudiants ayant obtenu de notes pour toutes les matières
24) Créer une table contenant la liste des admis (moyenne >=10) et une autre table pour les des redoublants