

# TP1 – BD Relationnelles

## MySQL

**Dr FOTSING TALLA Bernard**

*PhD in Computer Science*

*Chargé de Cours, IUTFV de Bandjoun*

Université de Dschang

[bernard.fotsing@univ-dschang.org](mailto:bernard.fotsing@univ-dschang.org)

Version 2.0 – Octobre 2019

---

# Objectifs du TP

- Appliquer les concepts théoriques étudiés en BD relationnelle sur le SGBD MySQL
- L'étudiant doit pouvoir analyser un problème concret de Système d'Information,
- Utiliser le modèle Entité-Association (MCD) pour décrire le schéma conceptuel de la BD associée
- Utiliser le modèle relationnel (MLD) pour décrire le schéma logique de la BD
- Utiliser MySQL pour créer le schéma physique de la BD
- Utiliser le langage de requête SQL pour peupler et interroger la BD

# Plan

- 1 Prise en main de MySQL**
- 2 Créer et utiliser une BD**
- 3 Créer les tables**
- 4 Insérer les données dans les tables**
- 5 Exécuter les requêtes**

---

# 1 Prise en main de MySQL

- MySQL est un **Système de Gestion de Bases de Données Relationnelles** (SGBDR) basé sur le modèle **client-serveur**.
- C'est-à-dire que la BD se trouve sur un **serveur**, et pour interagir avec cette BD, il faut utiliser un **logiciel "client"** qui va interroger le serveur et transmettre la réponse que le serveur lui aura donnée
- Le serveur peut être installé sur une machine différente du client : situation réelle en entreprise où les BD sont très importantes
- Le serveur et le client peuvent être également installés sur la même machine : cas d'école.
- Dans tous les cas, chaque requête (insertion/modification/lecture de données) est faite par l'intermédiaire du client. Jamais vous ne discuterez directement avec le serveur
- Le **langage SQL** est utilisé pour la communication entre le client et le serveur. MySQL utilise aussi le langage SQL

---

# 1 Prise en main de MySQL

- **MySQL** un des SGBDR les plus utilisés.
- Sa popularité est due en grande partie au fait que c'est un logiciel open source, ie son code source est librement disponible et que quiconque peut modifier MySQL pour l'améliorer ou l'adapter à ses besoins.
- Une version gratuite de MySQL est par conséquent disponible.
- Et une version commerciale payante existe également
- Le développement de MySQL commence en 1994 par David Axmark et Michael Widenius.
- En 1995, la société MySQL AB est fondée par ces deux développeurs, et Allan Larsson. La première version officielle de MySQL sort la même année
- En 2008, MySQL AB est rachetée par la société Sun Microsystems, qui est elle-même rachetée par Oracle Corporation en 2010. Ce dernier édite aussi Oracle Database, un puissant SGBD qui est payant (et très cher).
- Mais Oracle a promis de continuer à développer MySQL et de conserver la double licence GPL (libre) et commerciale. Mais jusqu'à quand ?

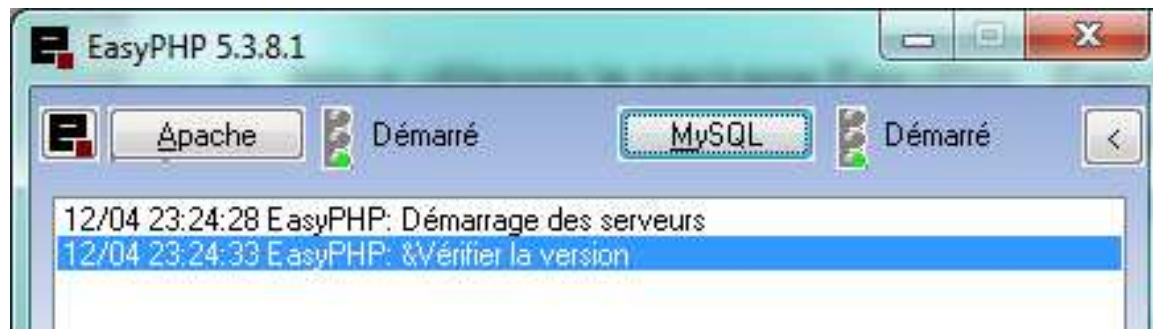
---

# 1 Prise en main de MySQL

- On peut utiliser MySQL en **ligne de commande** ou en **interface graphique**
- Comme interface graphique pour MySQL, on peut citer MySQL Workbench, PhpMyAdmin (souvent utilisé pour créer un site web en combinant MySQL et PHP) ou MySQL Front
- Nous préférons la ligne de commande :
  - La ligne de commande permet de faire des choses subtiles et compliquées, et parfois on doit obligatoirement écrire soi-même des requêtes ;
  - Ensuite, parce que vous aurez à utiliser MySQL en combinaison avec un autre langage de programmation. Or, dans du code PHP (ou Java, ou Python, etc.), on ne va pas écrire *"Ouvre PhpMyAdmin et clique sur le bon bouton pour que j'insère une donnée dans la base"*. On va écrire en dur les requêtes. Il faut donc que vous sachiez comment faire **dès maintenant**.
- Si vous avez déjà installé un logiciel qui permet de gérer une base de données MySQL, typiquement PHPMyAdmin qui est contenu dans les célèbres MAMP (pour Mac) et WAMP (pour Windows), ou EasyPHP, il n'est pas nécessaire de réinstaller MySQL. Il faut juste le localiser.

# 1 Prise en main de MySQL

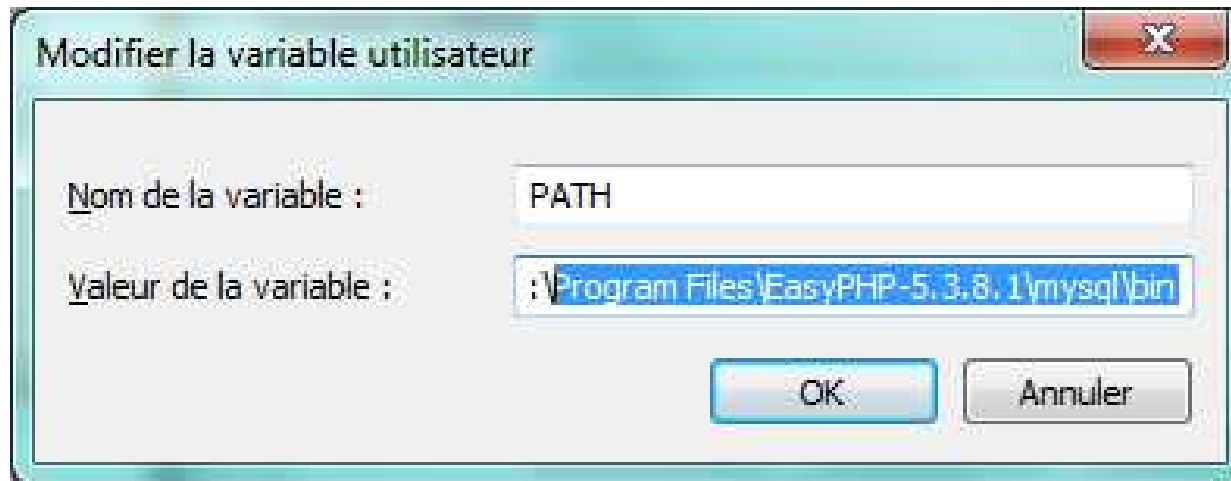
- Téléchargez MySQL ici <https://dev.mysql.com/downloads/mysql/#downloads>
- Nous utilisons le package EasyPhp : EasyPHP-5.8.1 qui contient PHP 5.8 VC9, Apache 2.2.21 VC9, MySQL 5.5.16, PhpMyAdmin 4.5, Xdebug 2.1.2
- A la fin de l'installation (en choisissant les valeurs par défaut), Vérifier que MySQL est démarré en faisant **Menu Démarrer → Tous les Programmes → EasyPHP 5.8.1 → EasyPHP 5.8.1.**



- S'il n'est pas démarré, cliquer sur le bouton correspondant pour le faire
- EasyPHP installe un produit pour l'administration d'une base de données MYSQL appelé PhpMyAdmin.

# 1 Prise en main de MySQL

- Pendant l'installation, un utilisateur par défaut **root** avec/sans mot de passe
- Modifier la variable d'environnement Path en ajoutant le chemin absolu où est installé mysql : C:\Program Files\EasyPHP-5.8.1\mysql\bin



- Ou en ligne de commande par  
`set PATH=%PATH%; C:\\\"Program Files\"\\EasyPHP-5.8.1\\mysql\\bin`
- Démarrer ensuite l'invite de commande DOS en cliquant sur le menu **Démarrer → Exécuter**. Puis taper la commande **cmd**.

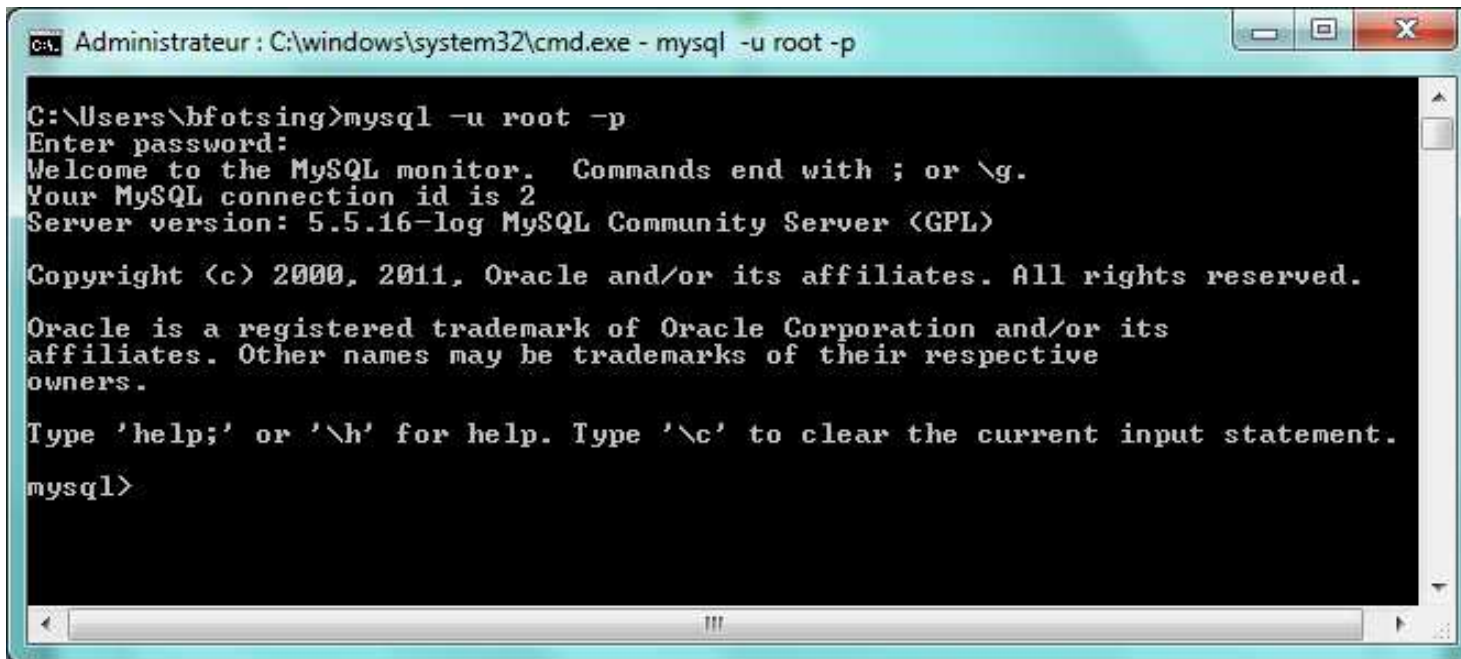


# 1 Prise en main de MySQL

- Pour se connecter, le client *mysql* a besoin d'au minimum trois paramètres :
  - l'hôte : c'est-à-dire l'endroit où est localisé le serveur ;
  - le nom d'utilisateur ;
  - et le mot de passe de l'utilisateur.
- Pour chacun des trois paramètres, la syntaxe est :
  - ***--hote=nom\_hote*** ou ***-h nom\_hote***
  - ***--user=nom\_utilisateur*** ou ***-u nom\_utilisateur***
  - ***--password=password*** ou ***-ppassword***
- Quelques exemples de connexion
  - ***mysql -h localhost -u root -p motdepasse topsecret***
  - ***mysql --host=localhost --user=root --password=motdepasse topsecret***
  - ***mysql -h localhost --user=root -p motdepasse topsecret***
  
  - ***mysql -h localhost -u root -p*** ou simplement ***mysql -u root -p***
  - ***Enter password :***
- Pour se déconnecter du client, on utilise la commande ***quit*** ou ***exit***

# 1 Prise en main de MySQL

- Pour se connecter à MySQL, taper la commande : ***mysql -u root -p***. Ne tapez rien pour le mot de passe et validez



```
Administrateur : C:\windows\system32\cmd.exe - mysql -u root -p

C:\Users\bfotsing>mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.5.16-log MySQL Community Server (GPL)

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql>
```

- Taper aussi ***Select 'Hello World';***
- Taper ***help*** ou ***\h*** pour avoir de l'aide sur les commandes de MySQL.

# 1 Prise en main de MySQL

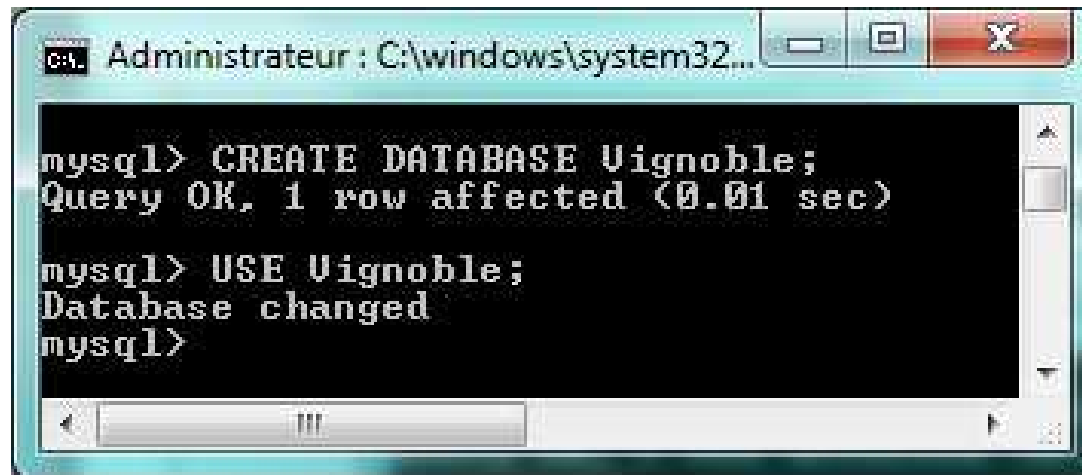
- En SQL, on utilise les `--` (double tirets) pour les commentaires. MySQL autorise aussi `#`
- En SQL et dans la plupart des SGBD, on entoure les chaînes de caractères par `'` (guillemets simples). MySQL autorise aussi les guillemets doubles `"`
- En SQL on échappe les caractères spéciaux (simple guillemet `\'`, tabulation `\t`, retour à la ligne `\n`, pourcent `\%`, souligné `\_`, antislash `\\`) par une simple guillemet. MySQL autorise aussi l'antislash `\`
- Lorsqu'on se connecte à MySQL, il est nécessaire de préciser l'encodage utilisé : soit pendant la connexion avec l'option **`--default-character-set`** ; soit avec la commande **`set NAMES`**.
- Il y a plusieurs jeux de caractères, le plus utilisé et recommandé est **`utf8`** qui de représenter un plus grand nombre de caractères (plus de 65 000) de plusieurs jeux de caractères différents
  - **`mysql -u student -p --default-character-set=utf8`**
  - **`SET NAMES 'utf8';`**

# 1 Prise en main de MySQL

- MySQL définit plusieurs types de données :
  - Des numériques entiers : *TINYINT*, *SMALLINT*, *MEDIUMINT*, *INT*, *BIGINT*
  - Des numériques décimaux : *NUMERIC* et *DECIMAL*, *FLOAT*, *DOUBLE* et *REAL*
  - Des textes alphanumériques : *CHAR* et *VARCHAR*, *TINYTEXT*, *TEXT*, *MEDIUMTEXT* et *LONGTEXT*
  - Des chaînes binaires alphanumériques : *BINARY* et *VARBINARY*, *TINYBLOB*, *BLOB*, *MEDIUMBLOB* et *LOB*
  - Des chaînes énumérées alphanumériques : *SET* et *ENUM*
  - Des données temporelles : *DATE*, *TIME*, *DATETIME*, *TIMESTAMP*, *YEAR*
- Il faut toujours utiliser le type de données adapté à votre situation
- Les types *SET* et *ENUM* sont des types de données qui n'existent que chez MySQL. Il faut donc les éviter.
- Le format de *DATE* est 'AAAA-MM-JJ'. N'importe quelle ponctuation peut être utilisée pour délimiter les parties (ou aucune)
- MySQL supporte des *DATE* allant de '1001-01-01' à '9999-12-31'
- MySQL stocke un *DATETIME* au format 'AAAA-MM-JJ HH:MM:SS'
- A ce niveau, l'environnement est prêt à recevoir des ordres SQL

## 2 Créer, utiliser et supprimer une BD

- Pour créer une BD, taper la commande : **CREATE DATABASE** <nom\_bd>;
- Ou **CREATE DATABASE** <nom\_bd> **CHARACTER SET** 'utf8';
- Chaque fois qu'on se connecte à MySQL, il faut sélectionner la BD dans laquelle on va travailler avec la commande : **USE** <nom\_bd> ;



```
mysql> CREATE DATABASE Vignoble;
Query OK, 1 row affected (0.01 sec)

mysql> USE Vignoble;
Database changed
mysql>
```

- On peut se connecter en spécifiant la BD par :
  - **mysql -u bft -p vignoble**
- Pour supprimer une BD, taper **DROP DATABASE** <nom\_bd> ; ou encore **DROP DATABASE IF EXISTS** <nom\_bd> ;

## 3 Créer les tables

- Les **moteurs de tables** sont une spécificité de MySQL. Ce sont des moteurs de stockage. Cela permet de gérer différemment les tables selon l'utilité que l'on en a.
- Les deux moteurs les plus connus sont **MyISAM** et **InnoDB**. **MyISAM** est le moteur par défaut
- **Avantages de MyISAM** : les commandes d'insertion et sélection de données sont particulièrement rapides sur les tables utilisant ce moteur.
- **Inconvénients de MyISAM** : il ne gère pas certaines fonctionnalités importantes comme les clés étrangères, qui permettent de vérifier l'intégrité d'une référence d'une table à une autre table ou les transactions, qui permettent de réaliser des séries de modifications "en bloc", ou au contraire d'annuler ces modifications
- **Avantages de InnoDB** : il gère les clés étrangères et les transactions. En cas de crash du serveur, il possède un système de récupération automatique des données.
- **Inconvénients de InnoDB** : Plus lent et plus gourmand en ressources que **MyISAM**

## 3 Créer les tables

- Pour qu'une table utilise le moteur de notre choix, il suffit d'ajouter à la fin de la commande de création l'instruction **ENGINE=moteur**;
- La syntaxe de création d'une table est :

```
CREATE TABLE [IF NOT EXISTS] Nom_table (  
    colonne1 description_colonne1,  
    [colonne2 description_colonne2,  
    colonne3 description_colonne3,  
    ...,]  
    [PRIMARY KEY (colonne_clé_primaire)]  
) [ENGINE=moteur];
```

- Lorsque IF NOT EXISTS est présente et si une table de ce nom existe déjà dans la BD, la requête renvoie un warning plutôt qu'une erreur.
- On peut aussi ne pas préciser la clé primaire directement à la création de la table. Il est tout à fait possible de l'ajouter par la suite

## 3 Créer les tables

- Considérons la table ***Etudiant*** avec ses six colonnes suivantes

Caractéristique	Nom du champ	Type	NULL?	Divers
Numéro d'identité	id	SMALLINT	Non	Clé primaire + auto-incrément +UNSIGNED
Matricule	matricule	VARCHAR(20)	Non	-
Nom	Nom	VARCHAR(30)	Oui	-
Sexe	Sexe	CHAR(1)	Oui	Exclusivement F ou M, par défaut F
Date de naissance	date_nais	DATETIME	Non	-
Description	description	TEXT	Oui	-

- En utilisant l'auto-incrémentation pour la colonne id, on demande à MySQL d'ajouter une valeur fixée qui est automatiquement incrémenté de 1 à chaque nouvelle inscription sans intervention du programmeur.



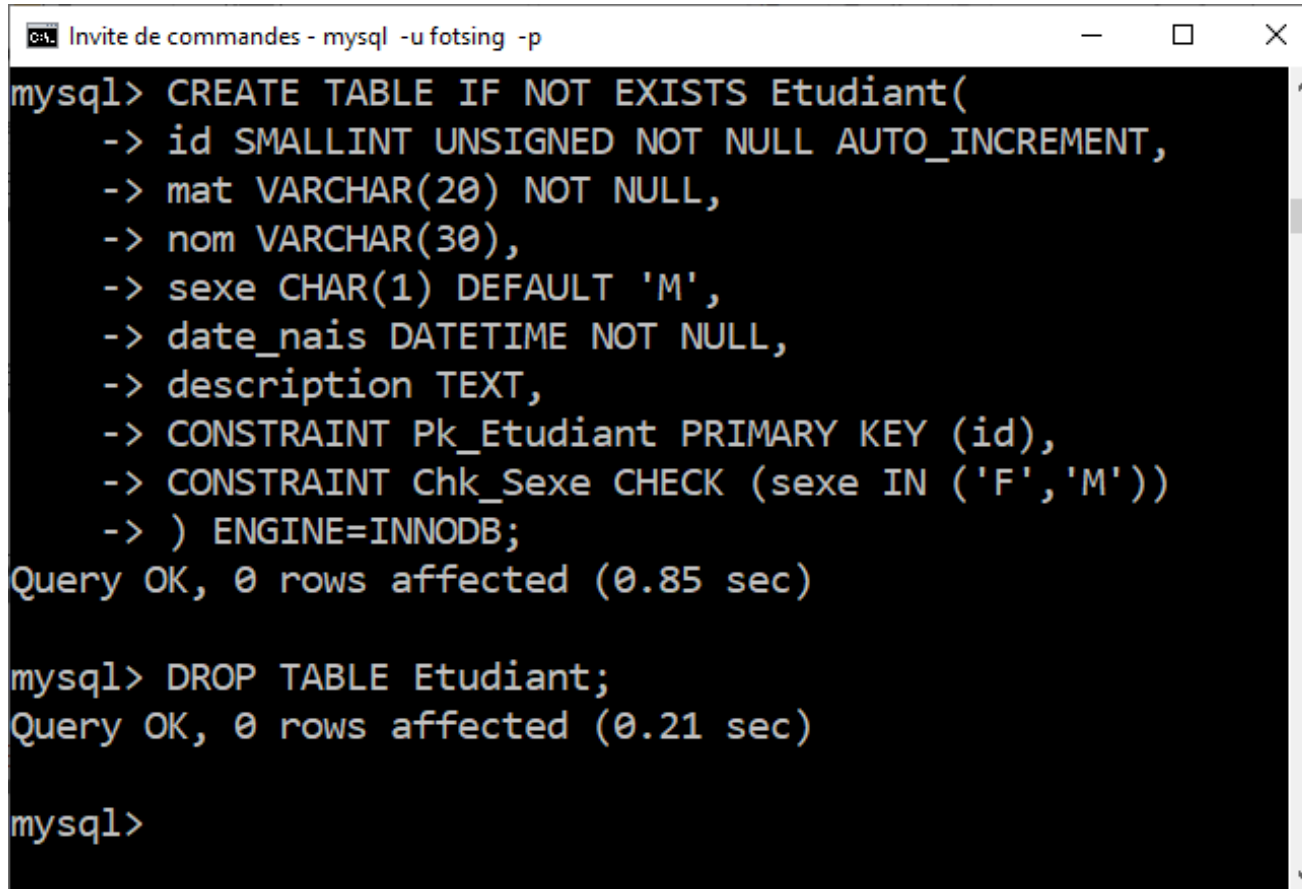
## 3 Créer les tables

- La syntaxe de création d'une table est :

```
CREATE TABLE Etudiant (  
    id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,  
    mat VARCHAR(20) NOT NULL,  
    nom VARCHAR(30),  
    sexe CHAR(1) DEFAULT 'F',  
    date_nais DATETIME NOT NULL,  
    description TEXT,  
    CONSTRAINT Pk_Etudiant PRIMARY KEY (id),  
    CONSTRAINT Chk_Sexe CHECK (sexe IN ('F', 'M')),  
)  
ENGINE=INNODB;
```

- Pour supprimer une table on utilise la commande DROP  
**DROP TABLE** Etudiant ;

### 3 Créer les tables



```
mysql> CREATE TABLE IF NOT EXISTS Etudiant(  
  -> id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,  
  -> mat VARCHAR(20) NOT NULL,  
  -> nom VARCHAR(30),  
  -> sexe CHAR(1) DEFAULT 'M',  
  -> date_naiss DATETIME NOT NULL,  
  -> description TEXT,  
  -> CONSTRAINT Pk_Etudiant PRIMARY KEY (id),  
  -> CONSTRAINT Chk_Sexe CHECK (sexe IN ('F','M'))  
  -> ) ENGINE=INNODB;  
Query OK, 0 rows affected (0.85 sec)  
  
mysql> DROP TABLE Etudiant;  
Query OK, 0 rows affected (0.21 sec)  
  
mysql>
```

- Avant de supprimer la table, taper **SHOW TABLES** ; et puis **DESCRIBE Etudiant** ;

# 3 Modifier la structure d'une table

- La commande ALTER TABLE permet de modifier la structure d'une table
- Lorsque l'on ajoute ou modifie une colonne, il faut toujours préciser sa (nouvelle) description **complète** (type, valeur par défaut, auto-incrément)  
*ALTER TABLE nom\_table ADD ... -- permet d'ajouter quelque chose (une colonne)*  
*ALTER TABLE nom\_table DROP ... -- permet de retirer quelque chose*  
*ALTER TABLE nom\_table CHANGE ... -- permet de changer le nom d'une colonne*  
*ALTER TABLE nom\_table MODIFY ... -- permettent de modifier une colonne*
- Soit la table de test suivante
- **CREATE TABLE Test ( id INT NOT NULL, nom VARCHAR(10) NOT NULL, PRIMARY KEY (id) );**
- **ALTER TABLE Test ADD COLUMN date\_insertion DATE NOT NULL;**
- **ALTER TABLE Test DROP COLUMN date\_insertion;**
- **ALTER TABLE Test CHANGE nom prenom VARCHAR(10) NOT NULL;**
- **ALTER TABLE Test MODIFY id BIGINT NOT NULL AUTO\_INCREMENT;**
- **ALTER TABLE Test MODIFY nom VARCHAR(30) NOT NULL DEFAULT 'Blabla';**
- **ALTER TABLE Test ADD CONSTRAINT Pk\_Test PRIMARY KEY (id);**

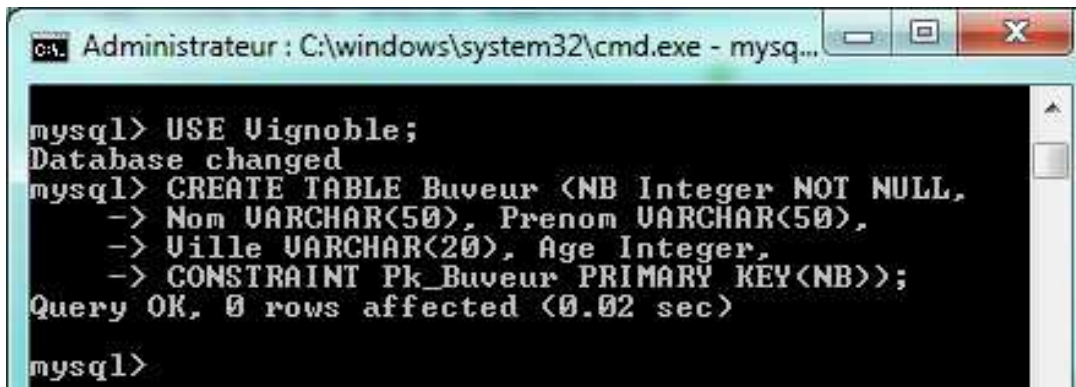
## 3 Créer les tables

*Buveur (NB, nom, prenom, ville, age)*

*Vins (NV, cru, region, millésime, degre)*

*Boire (NB#, NV#, date, quantité)*

- **CREATE TABLE** Buveur (NB Integer **NOT NULL**, Nom VARCHAR(50) , Prenom VARCHAR(50), Ville VARCHAR (20), Age Integer, **CONSTRAINT** Pk\_Buveur **PRIMARY KEY** (NB));



```
Administrateur : C:\windows\system32\cmd.exe - mysql...
mysql> USE Vignoble;
Database changed
mysql> CREATE TABLE Buveur (NB Integer NOT NULL,
-> Nom VARCHAR(50), Prenom VARCHAR(50),
-> Ville VARCHAR(20), Age Integer,
-> CONSTRAINT Pk_Buveur PRIMARY KEY(NB));
Query OK, 0 rows affected (0.02 sec)

mysql>
```

- Procéder de la même façon pour créer les deux autres tables

## 4 Insertion des données

- On a deux possibilités s'offrent si l'on veut insérer une ligne dans une table :
  - soit donner une valeur pour chaque colonne de la ligne,
- **INSERT INTO** Etudiant **VALUES** (1, '68IUT', 'Kougang', 'F', '2004-04-05 13:43:00', 'Major de la promotion');
- **INSERT INTO** Etudiant **VALUES** (**NULL**, '69IUT', 'FOTSO', **NULL**, '2000-03-24 02:23:00', **NULL**);
  - soit ne donner les valeurs que de certaines colonnes, auquel cas il faut bien sûr préciser de quelles colonnes il s'agit.
- **INSERT INTO** Etudiant(nom, sexe, date\_nais) **VALUES** ('Djuidje', 'F', '2009-08-03 05:12:00');
- **INSERT INTO** Etudiant (nom, description, date\_nais) **VALUES** ('Choupi', 'Né sans oreille gauche', '2010-10-03 16:44:00');
- **INSERT INTO** Etudiant (nom, sexe, date\_nais,) **VALUES**  
('Aissatou', 'F', '2008-12-06 05:18:00'),  
('Ondoua', 'M', '2008-09-11 15:38:00'),  
('La tortue', **NULL**, '2010-08-23 05:18:00');

## 4 Insertion des données

- Insérer dans chaque 5 enregistrements : Attention aux différentes contraintes (valeurs non nulles, clés primaires, clefs étrangères, etc.)



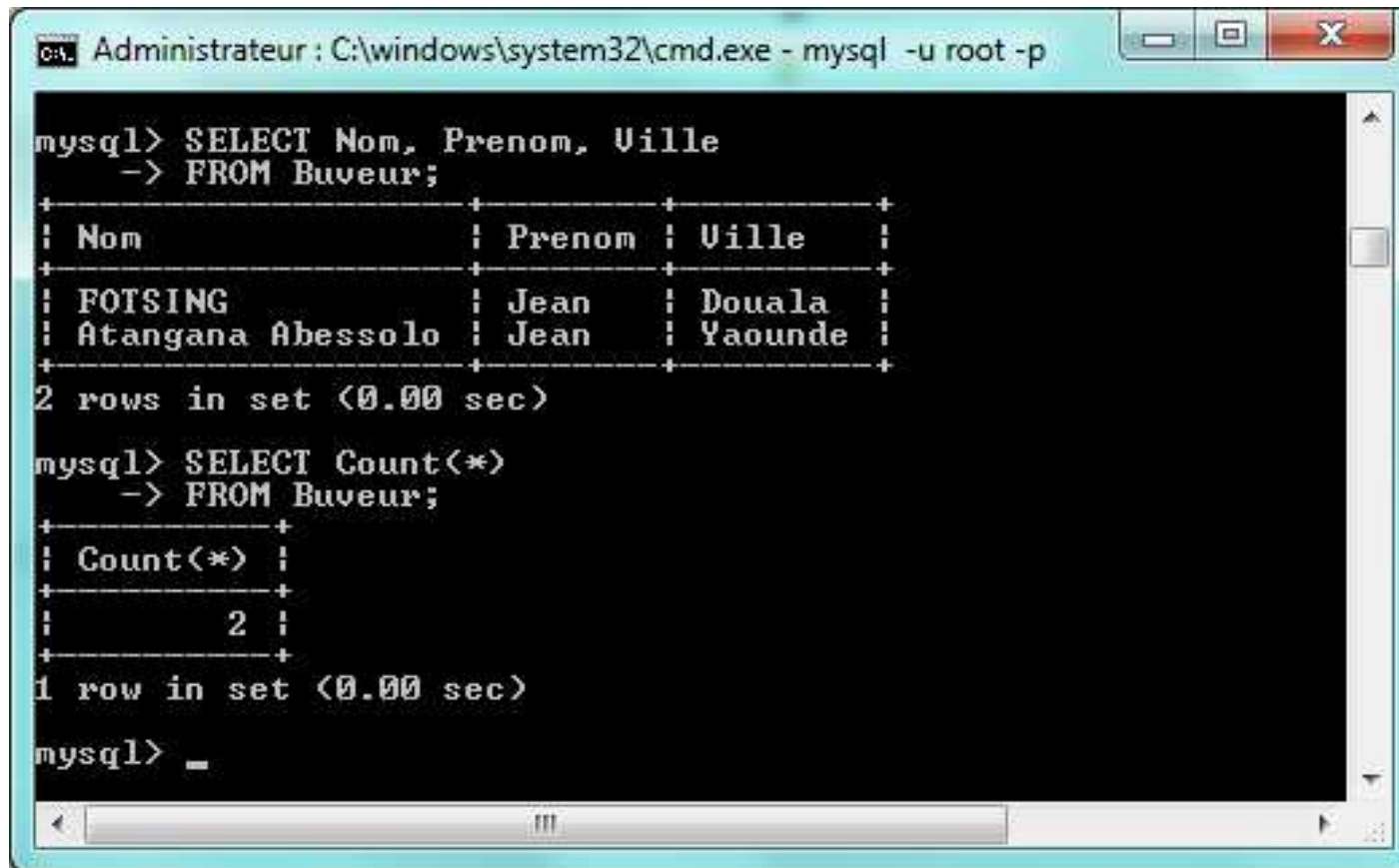
```
ca. Administrateur : C:\windows\system32\cmd.exe - mysql -u root -p

mysql> INSERT INTO Buveur (NB,Nom,Prenom,Ville,Age)
      -> VALUES(10,"Atangana Abessolo", "Jean", "Yaounde",25);
Query OK, 1 row affected (0.00 sec)

mysql> _
```

## 5 Exécuter des requêtes

- Tester toutes les requêtes vues en cours



```
Administrateur : C:\windows\system32\cmd.exe - mysql -u root -p

mysql> SELECT Nom, Prenom, Ville
-> FROM Buveur;
+-----+-----+-----+
| Nom      | Prenom | Ville  |
+-----+-----+-----+
| FOTSING  | Jean   | Douala |
| Atangana Abessolo | Jean   | Yaounde |
+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> SELECT Count(*)
-> FROM Buveur;
+-----+
| Count(*) |
+-----+
|         2 |
+-----+
1 row in set (0.00 sec)

mysql> _
```

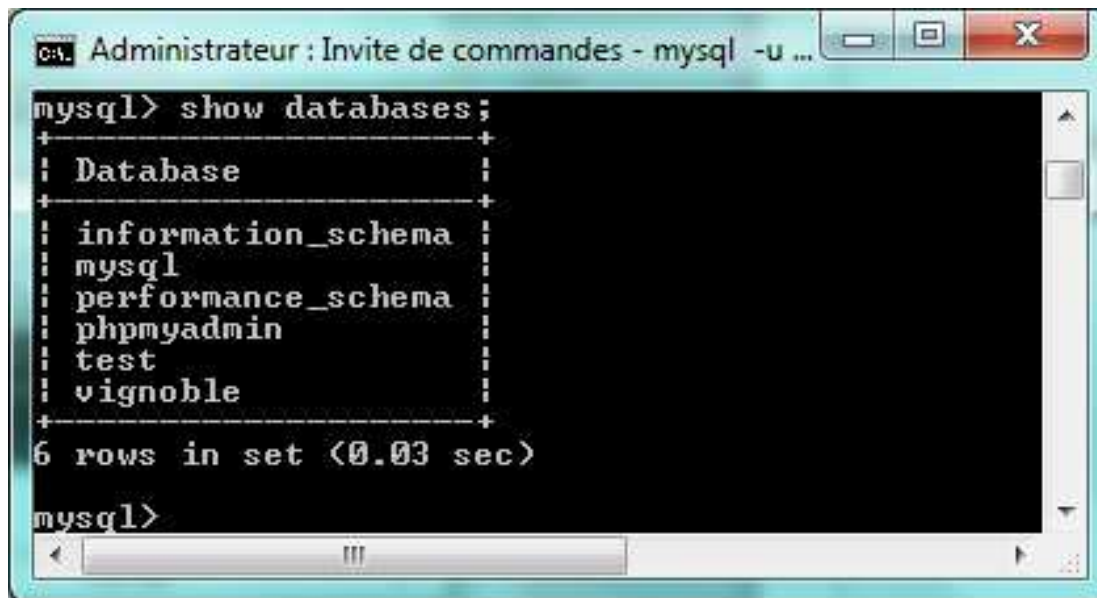
## 5.1 Quelques commandes usuelles

- ***SELECT DATABASE();*** C'est une pseudo-requête SQL (sans FROM) qui Affiche le nom de la base courante.
- ***SELECT USER();*** Idem, cette pseudo-requête affiche le nom de l'utilisateur courant.
- • ***SHOW DATABASES;*** Affiche la liste des bases de données.
- • ***SHOW TABLES;*** Affiche la liste des tables de la base courante.
- • ***SHOW COLUMNS FROM* <Nom-Table>;** Affiche la description de la table Nom-Table.
- ***SHOW GRANTS FOR* <utilisateur>**
- ***SHOW WARNINGS;*** afficher les avertissements sous MySQL
  
- Voir le schéma d'une table : DESCRIBE ou DESC
- *mysql> Use vignoble;*
- *mysql> DESC Buveur;*



## 5.1 Quelques commandes usuelles

- Montrer les différentes bases de données existantes
- *Mysql> **show Databases;***



```
Administrator : Invite de commandes - mysql -u ...
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql          |
| performance_schema |
| phpmyadmin     |
| test          |
| vignoble      |
+-----+
6 rows in set (0.03 sec)

mysql>
```

- Voir les tables d'une BD (Vignoble par exemple)
- *mysql> **Use vignoble;***
- *mysql> **Show tables ;***

## 5.1 Quelques commandes usuelles

- Créer un Nouvel Utilisateur (Avec GRANT ou CREATE USER (USAGE lui donne la possibilité de se connecter sans mot de passe) :
- `mysql> CREATE USER 'bfotsing'@'localhost' IDENTIFIED BY 'bft'`
- `mysql> GRANT USAGE ON Vignoble.* TO bfotsing ;`
- Ou directement avec tous les privilèges :
- `mysql> GRANT ALL PRIVILEGES ON Vignoble.* TO 'fotsing'@'localhost' IDENTIFIED BY 'bernard';`



```
Administrateur : Invite de commandes - mysql -u fotsing -p
mysql> GRANT ALL PRIVILEGES ON Vignoble.* TO fotsing@localhost IDENTIFIED BY 'bernard';
Query OK, 0 rows affected (0.02 sec)

mysql> Commit;
Query OK, 0 rows affected (0.00 sec)

mysql> exit;
Bye

C:\Users\bfotsing>mysql -u fotsing -p
Enter password: *****
```

---

## 5.1 Quelques commandes usuelles

- `mysql> REVOKE create, drop ON *.* FROM bfotsing@localhost;`  
Retire à bfotsing@localhost le droit de créer ou de détruire des tables.
- `mysql> SET PASSWORD FOR visiteur@localhost =  
PASSWORD('monMot');`  
Affecte un mot de passe à l'utilisateur visiteur

---

## 5.2 Fichiers .sql et .cvs

- Il est fastidieux d'entrer au clavier toutes les commandes SQL en ligne de commande mysql, d'autant que toute erreur de frappe implique de recommencer la saisie au début.
- Une meilleure solution est de créer un fichier (.sql) contenant les commandes et de l'exécuter: on parle de script SQL.
- Un script SQL peut contenir tout un ensemble de commandes, chacune devant se terminer par un ';'. Toutes les lignes commençant par « # », ou tous les textes encadrés par /\*,\*/, sont des commentaires.
- On indique à mysql qu'il doit prendre ses commandes dans ce fichier au lieu de l'entrée standard de la manière suivante :
- ***C:> mysql -u adminFilms -p < MonScriptSql.sql***
- ou simplement, si on utilise un fichier de configuration avec nom et mot de passe : ***C:> mysql < MonScriptSql.sql***
- Le caractère « < » permet une redirection de l'entrée standard (par défaut la console utilisateur) vers *MonScriptSql.sql*.

---

## 5.2 Fichiers .sql et .csv

- Dernière solution, quand on est déjà sous l'interpréteur de MySQL, on peut exécuter les commandes contenues dans un fichier avec la commande ***source*** ou ***\.***
- ***mysql> source MonScriptSql.sql***
- ***mysql> \. MonScriptSql.sql***
- La commande ***LOAD DATA [LOCAL] INFILE*** permet de charger des données dans une table à partir d'un fichier formaté (.csv par exemple)  
nom;mat;date\_nais  
Charles;70IUT;1994-12-30  
Bruno;72IUT;1978-05-12  
Mireille;73IUT;1990-08-23
- MySQL permet de lire ce type de fichier afin de remplir une table avec les données contenues dans le fichier.
- La syntaxe est la suivante :

---

## 5.2 Fichiers .sql et .cvs

```
LOAD DATA [LOCAL] INFILE 'nom_fichier'
INTO TABLE nom_table
[FIELDS
  [TERMINATED BY '\t']
  [ENCLOSED BY '"']
  [ESCAPED BY '\\']
]
[LINES
  [STARTING BY '"']
  [TERMINATED BY '\n']
]
[IGNORE nombre LINES]
[(nom_colonne,...)];
```

---

## 5.2 Fichiers .sql et .cvs

- Le mot-clé **LOCAL** sert à spécifier si le fichier se trouve côté client (dans ce cas, on utilise **LOCAL** ou côté serveur (auquel cas, on ne met pas **LOCAL**)
- Si le fichier se trouve du côté du serveur, il est obligatoire, pour des raisons de sécurité, qu'il soit dans le répertoire de la base de données
- Les clauses **FIELDS** (pour les colonnes) et **LINES** (pour les lignes) permettent de définir le format de fichier utilisé.
  - **TERMINATED BY** définit le caractère séparant les colonnes, entre guillemets
  - **ENCLOSED BY** définit le caractère entourant les valeurs dans chaque colonne (vide par défaut).
  - **ESCAPED BY** définit le caractère d'échappement pour les caractères spéciaux.
  - **STARTING BY** définit le caractère de début de ligne (vide par défaut)
  - **TERMINATED BY** définit le caractère de fin de ligne ('\\n' par défaut)
- La clause **IGNORE nombre LINES** permet d'ignorer un certain nombre de lignes. Par exemple, si la première ligne de votre fichier contient les noms des colonnes, vous ne voulez pas l'insérer dans votre table. Il suffit alors d'utiliser **IGNORE 1 LINES**

## 5.2 Fichiers .sql et .csv

- Exemple : créer un fichier etudiant.csv contenant les lignes suivantes. Attention, le fichier doit se terminer par un saut de ligne !

```
nom;mat;date_nais  
Charles;70IUT;1994-12-30  
Bruno;72IUT;1978-05-12  
Mireille;73IUT;1990-08-23
```

- Enregistrer le fichier sur votre DD dans un repertoire bien connu
- Exécutez la commande suivante.

```
LOAD DATA LOCAL INFILE 'etudiant.csv'  
INTO TABLE Etudiant  
FIELDS TERMINATED BY ';'   
LINES TERMINATED BY '\n'  
IGNORE 1 LINES  
(nom,mat,date_nais);
```





FIN