# BASE DE DONNEES MySQL

# Plan de formation

1. MCD et MLD
2. Base de données Mysql
3. Langage SQL
4. Sauvegarde et Restauration
5. Gérer les utilisateurs et leurs droits

# Plan de formation

1. MCD et MLD
2. Base de données Mysql
3. Langage SQL
4. Sauvegarde et Restauration
5. Gérer les utilisateurs et leurs droits

# MCD et MLD

## Exemple de données sur un bon de commande:

Date Commande: **05/05/2014**

Commande N°: **2014000145**
Client Numero: **1452**
Nom Client: **RANAIVO Paul**
Adresse Client: **ZI KAWENI**

| Numero Article | Designation | Prix Unitaire | Quantite | Montant |
|---|---|---|---|---|
| 1045 | Jus ananas 1l | 1,15 | 48 | 55,20 |
| 1015 | Eau 0,5l pack 6 | 2,10 | 12 | 25,20 |
| 1258 | Cahier | 3,00 | 10 | 30,00 |
| Total | | | | 110,40 |

**Recensement de données**

| code | Description | Type | Contrainte |
|---|---|---|---|
| dateCommande | Date Commande | date | |
| numeroCommande | Numero Commande | Entier | non Null |
| numeroClient | Numero Client | Entier | non Null |
| nomClient | Nom Client | Caractères(80) | |
| adresseClient | Adresse Client | Caractères(250) | |
| numeroArticle | Numero Article | Entier | non Null |
| designation | Designation | Caractères(150) | |
| prixUnitaire | Prix Unitaire | Reel | |
| quantite | Quantité commandée | Reel | |
| Montant | Montant | Reel | =quantite*prixUnitaire |
| Total | Total commande | | = ∑ Montant |

# MCD et MLD

## MCD

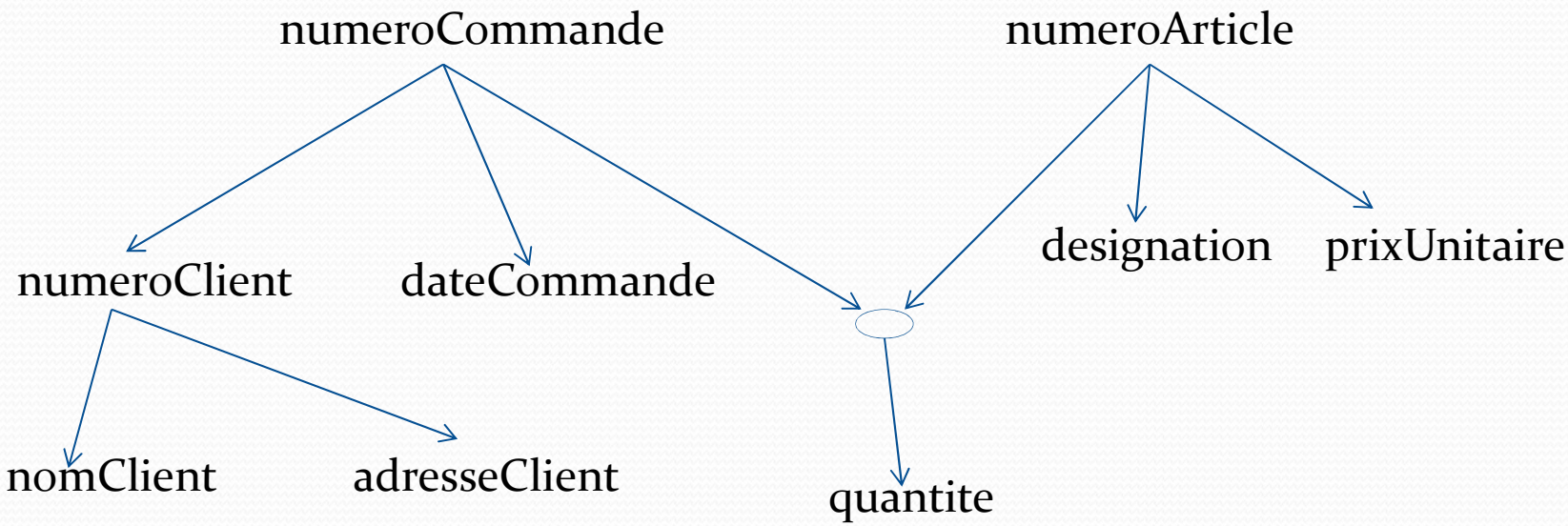
Le **M**odèle **C**onceptuel de **D**onnée (MCD) décrit de façon formelle les données qui seront utilisées par le système d'information.

Le MCD est exprimé en entité-relation et comporte les concepts basiques suivants :

- **Entité** : modélisation d'un objet d'intérêt (en terme de gestion) pour l'utilisateur.
- **Relation** : modélisation d'une association entre deux ou plusieurs entités
- **Cardinalités** : modélisation des participations mini et maxi d'une entité à une relation.
- **Propriétés** : modélisation des informations descriptives rattachées à une entité ou une relation.
- **Identifiant** : modélisation des propriétés contribuant à la détermination unique d'une occurrence d'une entité.

# MCD et MLD

## MCD : *Graphe de couverture minimale*



# MCD et MLD

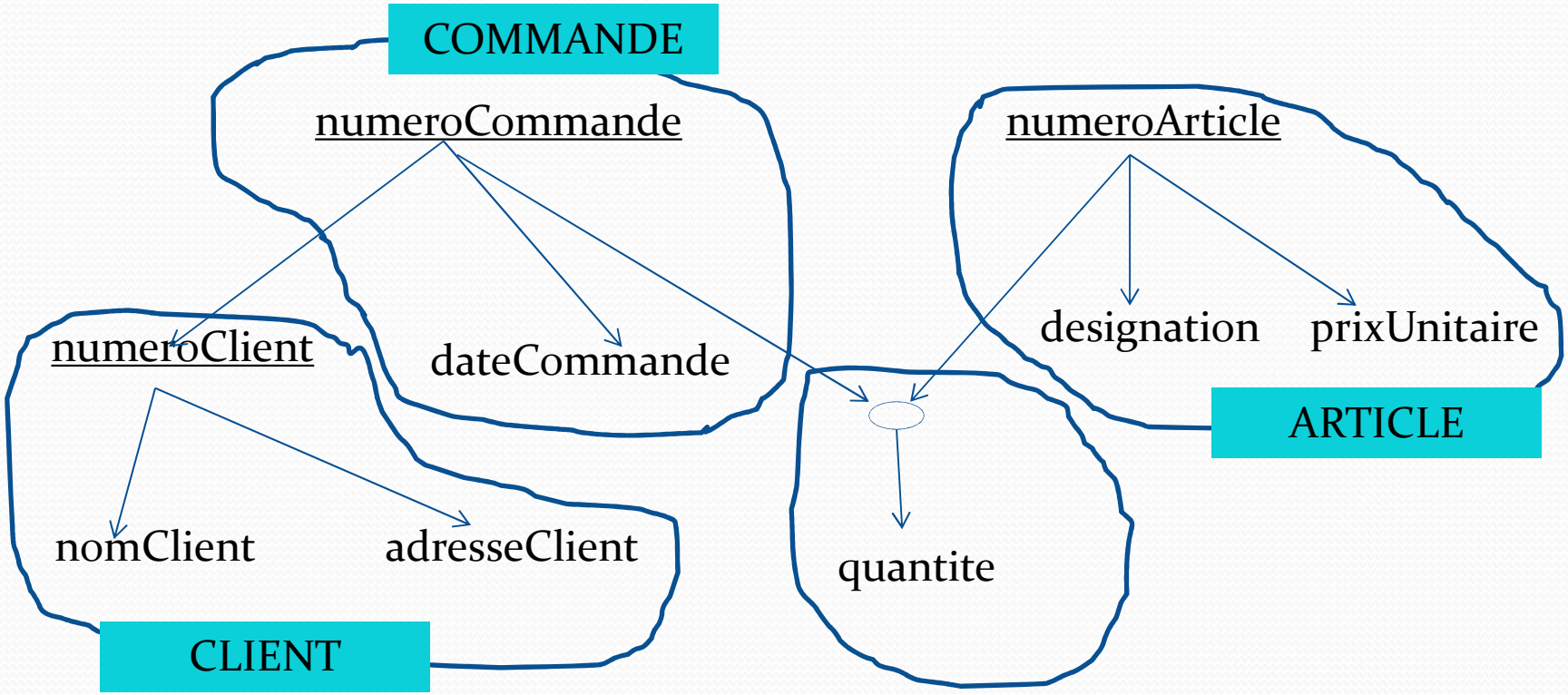
## MCD : *Traduction du graphe vers un schéma entites-associations*

A partir du graphe de couverture minimale, les étapes à suivre sont les suivantes:

- **Etape 1** : repérer et souligner les identifiants
- **Etape 2** : tous les attributs en dépendance directe à l'identifiant forme avec ce dernier une entité
- **Etape3:** les dépendances entre les identifiants forment des associations

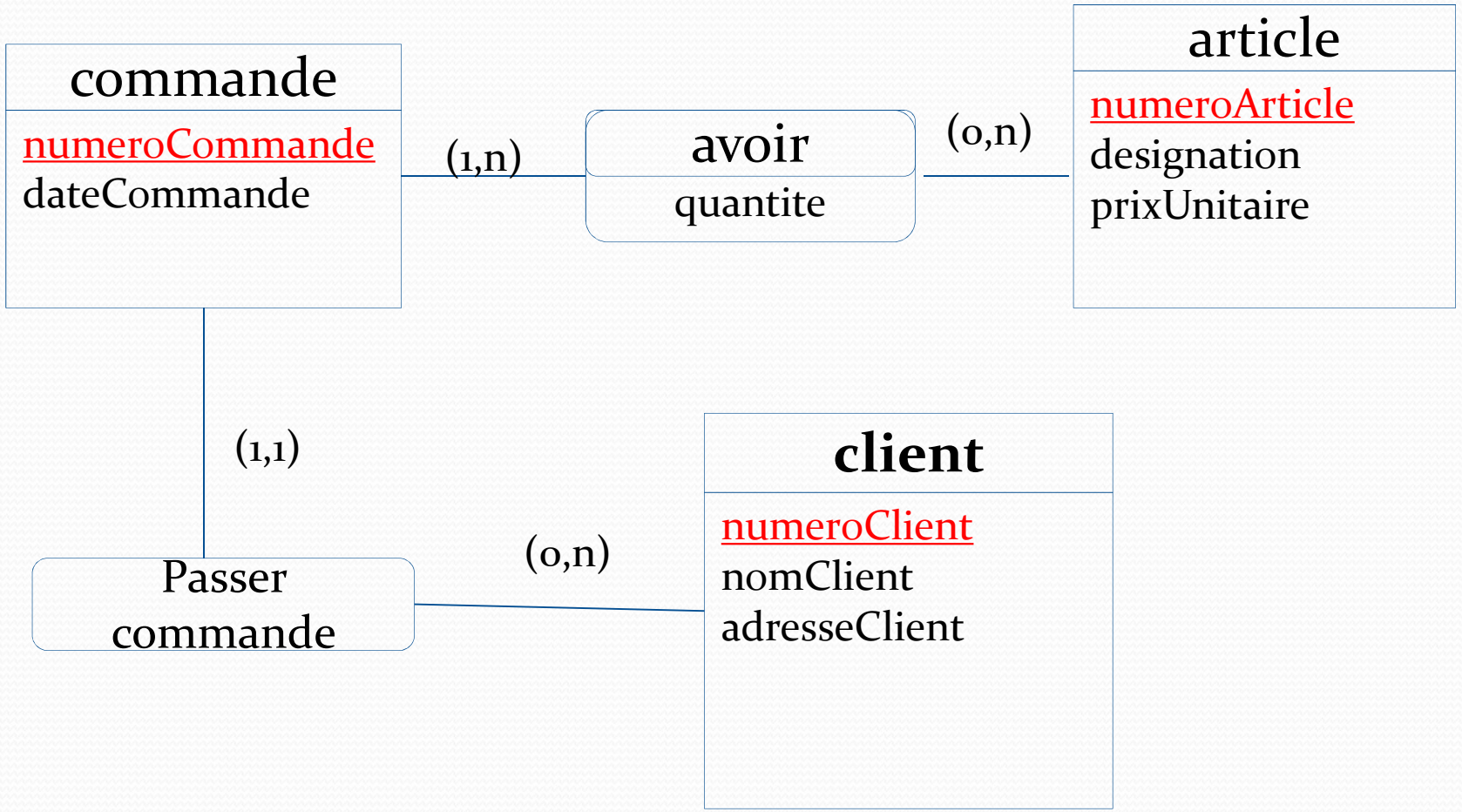
# MCD et MLD

## MCD



# MCD et MLD

## MCD : Modèle conceptuel de données ( MCD)



# MCD et MLD

## Règle de conversion MCD vers MLD

Le **M**odèle **L**ogique des **D**onnées (MLD) décrit la structure de données sans faire référence à un langage de programmation

## Règles basiques de transformation MCD vers MLD (1/2) :

- Entités :
  - Toute entité devient une table.
  - L'identifiant de l'entité devient une clé primaire de cette table.
  - Les propriétés de l'entité deviennent des attributs.

# MCD et MLD

**Règle de conversion MCD vers MLD : *Règles basiques de transformation MCD vers MLD (2/2) :***

- Pour les relations, cela dépend de la dimension et des cardinalités
  - ***Relation avec une cardinalité *,1 :***
    - La relation devient un lien référentiel avec une clé étrangère dans la table correspondant à l'entité coté cardinalité *,1
  - ***Relation avec des cardinalités *,n -*,n :***
    - La relation devient une table
    - La clé primaire de la table est composée des clés étrangères référant aux clés primaires des entités composant la relation.
    - Les éventuelles propriétés de la relation deviennent des attributs de la table.

# MCD et MLD

## Règle de conversion MCD vers MLD : *Schéma relationnel de MLD*

# MCD et MLD

## Clé primaire informatique sur le MLD

Pour optimiser la base de données, il est préférable de remplacer les clés primaires utilisateurs par des clés primaires informatique de type entier.

# MCD et MLD

## Représentation linéaire du MLD

**Commande ( ID, #client_ID, numeroCommande, dateCommande)**

**Client ( ID, numeroClient,nomClient, adresseClient)**

**LigneCommande (ID, #commande_ID, #article_ID, quantite)**

**Article (ID, numeroArticle,designation, prixUnitaire)**

# Plan de formation

1. MCD et MLD
2. Base de données Mysql
3. Langage SQL
4. Sauvegarde et Restauration
5. Gérer les utilisateurs et leurs droits

# Base de données Mysql

## Installation avec Linux

On utilise le gestionnaire de paquet qui va faire tout le travail pour nous. Pour cela on ouvre une console et on tape la commande suivante :

***sudo apt-get install*** *mysql-server*

Une fois l'installation achevée, on va vérifier qu'on possède bien la dernière version en tapant la commande suivante dans la console :

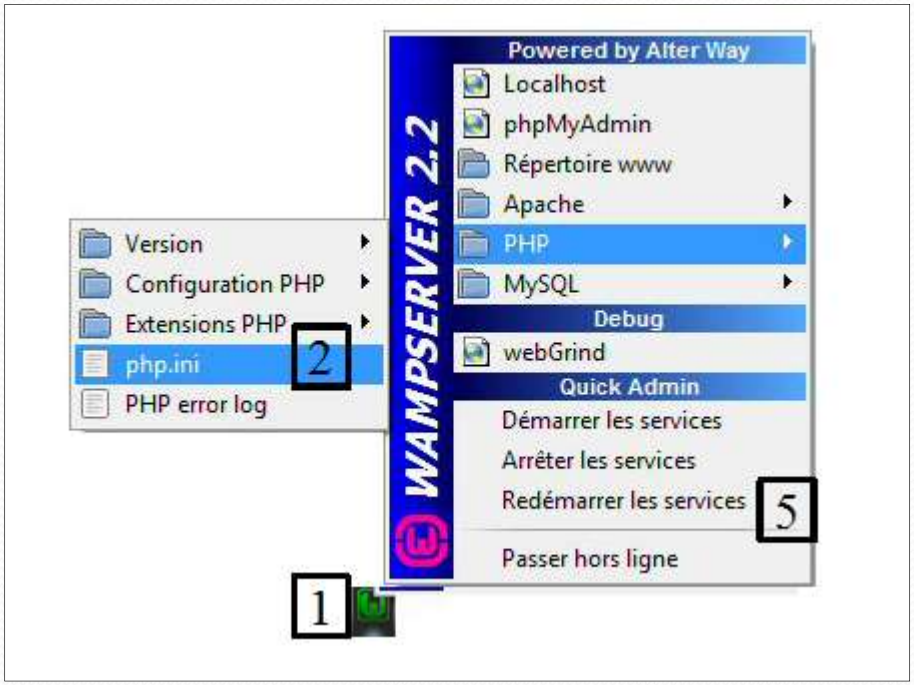
*mysql -V*

# Base de données Mysql

## Installation avec Windows

Pour l'installation sous Windows, vous pouvez telecharger le paquet WampServer (Windows ApacheMysql Php).

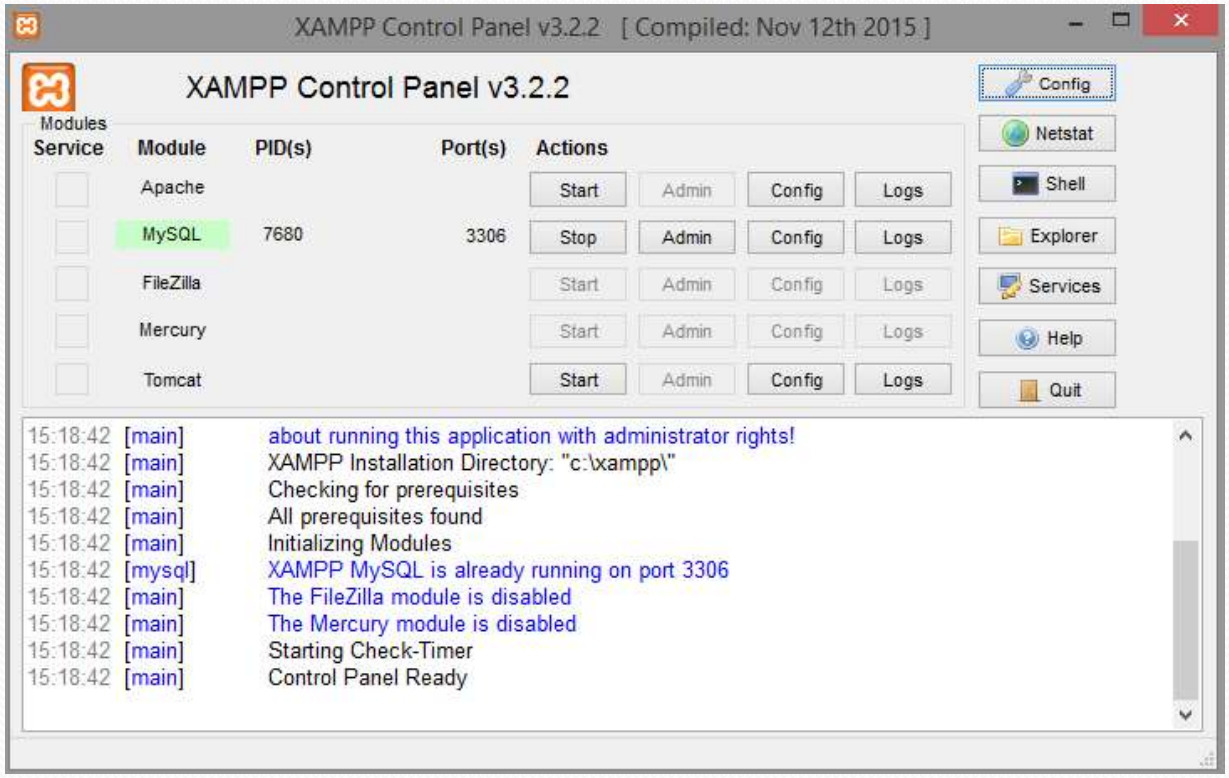
Après intallation vous devez avoir un icône vert représentant le WampServer dans la barre de tache



# Base de données Mysql

## Installation avec Windows

Vous pouvez télécharger aussi le XAMPP:



# Base de données Mysql

## Arrêter ou Redémarrer Mysql avec Lunix

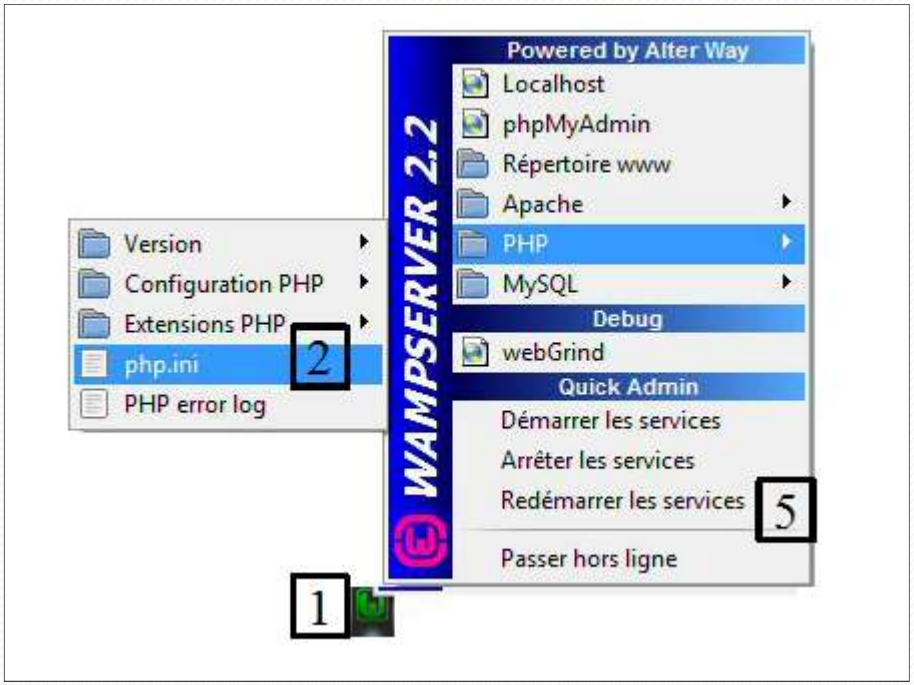
Après l'istallation Mysql est déjà démarré. Un des cas de l'arret et redémarrage est sur le  fait qu'on a fait quelques modifications sur les paramètres.

*/etc/init.d# service Mysql {start |restart| stop|reload }*

# Base de données Mysql

## Arrêter ou Redémarrer Mysql avec Windows

Cliquez sur l'icône vert de Wamp puis cliquer  sur 'Arreter les services'



# Base de données Mysql

## Entrer au serveur en mode console

**> Mysql –u root**

Vous arriverez alors sur un prompt de type :

**Mysql>**

## Entrer au serveur avec la base de données bdd_test

**> Mysql –u root –D bdd_test**

# Base de données Mysql

## Commandes internes (1/2)

| Commande | Description |
|---|---|
| ***Show databases;***<br><br>***Show tables;*** | -Affiche à l'écran les noms des bases de données existantes<br>-Affiche les noms des tables existantes |
| ***Connect nom_bdd;***<br>***ou***<br>***Use nom_bdd;***<br><br>***Select database();*** | -Se connecter à la base de données base_bdd<br>Exemple: connect dbb_e_commerce;<br><br>*-Affiche le nom de la base de donnée en cours* |
| | |
| **source *nomfichier***<br>***Ou***<br>**\. *nomfichier*** | Lit l'entrée à partir du fichier ***nomfichier*** et l'exécute comme si elle avait été saisie sur le clavier. |

# Base de données Mysql

## Commandes internes (2/2)

| Commande | Description |
|---|---|
| **Show index from nom_table** | Affiche les noms des index à la table **nom_table** |
| **tee** ***nomfichier*** <br><br> ***notee*** | Sauvegarde les résultats des requêtes dans le fichier **nomfichier** . <br> Termine le sauvegarde. |
| **Select * INTO OUTFILE 'chemin' from nom_table** | Export le résultat de la requête vers le fichier indiqué par le 'chemin' |
| **Desc nomTable** | Affiche la structure de la table **nomTable** . |

# Plan de formation

1. MCD et MLD
2. Base de données Mysql
3. Langage SQL
4. Sauvegarde et Restauration
5. Gérer les utilisateurs et leurs droits

# Langage SQL

## Les principaux types de langages

Le langage SQL comporte :

- une partie sur la définition des données :
  le **LDD (langage de définition des données )** qui permet de définir des tables, des vues et des contraintes d'intégrité (CREATE, ALTER,TRUNCATE, DROP, RENAME)
- une partie sur les requêtes :
  le **LMD (langage de manipulation des données)** qui permet d'interroger une base de données (SELECT, INSERT, UPDATE, DELETE)
- une partie sur le contrôle des données :
  le **LCD (langage de contrôle des données)** qui permet de contrôler la sécurité et les accès aux données.

  (GRANT,REVOKE)

# Langage SQL

## Créer une base de données

Vous pouvez créer une base de données en utilisant la commande CREATE DATABASE .

Par exemple , créons la base de données base_dd:

*CREATE DATABASE base_dd;*

## Se connecter à une base de données créés

Vous pouvez vous connecter à une base de données en utilisant la commande CONNECT ou USE .

Par exemple , avec la base de données base_dd qu'on vient de créer :

*CONNECT base_dd;*

ou

*USE base_dd;*

# Langage SQL

## Créer une table

Vous pouvez créer une table en spécifiant le nom de la table, suivi du nom de toutes les colonnes et de leur type .

Par exemple , créons la table nommée ***client***:

```
CREATE TABLE client(id integer auto_increment primary
   key ,
nomClient varchar(80), adresseClient varchar(150)
);
```

***Les options:***

- ✓ *Auto_incremente* permet d'incrémenter la valeur de l'attribut ***id*** à chaque insertion.
- ✓ *primary key* précise que ***id*** est la clé primaire de la table ***client***

# Langage SQL

## Clés étrangères (1/3)

Soient les tables **client** et **commande** . Il s'agit maintenant de s'assurer que personne n'insère de ligne dans la table commande qui ne corresponde à une entrée dans la table Client.

*CREATE TABLE commande( id integer auto_increment primary key, client_id integer , foreign key(client_id) references client(id), dateCommande date);*

A cet effet , ***toute insertion*** d'enregistrement dans la table commande provoquera une erreur si la valeur de l'attribut client_id n'existe pas dans la table client.

# Langage SQL

## Clés étrangères (2/3)

***Une clé étrangère*** peut aussi contraindre et référencer un groupe de colonnes. Il faut alors l'écrire sous forme d'une contrainte de table.

Exemple de syntaxe :

```
CREATE TABLE t1 (
a integer PRIMARY KEY,
b integer,
c integer,
FOREIGN KEY (b, c) REFERENCES autre_table (c1, c2)
);
```

Exemple**:**

```
CREATE TABLE commande_produits (
no_produit integer REFERENCES produits ON DELETE RESTRICT,
id_commande integer REFERENCES commandes ON DELETE CASCADE,
quantite integer , PRIMARY KEY (no_produit, id_commande));
```

# Langage SQL

## Clés étrangères (3/3)

***RESTRICT*** empêche la suppression d'une ligne référencée.

***CASCADE*** indique que, lors de la suppression d'une ligne référencée, les lignes la référençant doivent être automatiquement supprimées.

Si vous n'avez plus besoin d'une table ou que vous voulez la recréer différemment, vous pouvez

la supprimer en utilisant la commande suivante :

***DROP TABLE nom_table;***

# Langage SQL

## Remplir une table

L'instruction **INSERT** est utilisé pour remplir une table.

Prenons comme exemple la table client:

*INSERT INTO Client (nomClient, adresseClient) VALUES ('Joé','Saint-Nazaire'), ('Richard','Angres');*

Il est aussi possible de aussi remplir une Table par une autre table de même structure:

*INSERT INTO Client (nomClient, adresseClient) (Select nom,Adreese from Fournisseur);*

# Langage SQL

## Supprimer des enregistrements d'un table

L'instruction **DELETE** est utilisé pour supprimer des enregistrements.

Prenons comme exemple la suppression de tous les enregistrements de la table CLIENT dont adresseClient = 'Saint-Nazaire':

*DELETE FROM CLIENT where adresseClient='Saint-Nazaire';*

Il est aussi possible de vider complètement la table en utilisant la clause **TRUNCATE** :

*TRUNCATE TABLE CLIENT;*

# Tables jointes (1/2)

**Une table jointe** est une table dérivée de deux autres tables suivant les règles du type de jointure particulier

Soient t1 et t2 deux tables :

**Table t1**

| no | nom |
|---|---|
| 1 | a |
| 2 | b |
| 3 | c |

**Table t2**

| no | valeur |
|---|---|
| 1 | xxx |
| 3 | yyy |
| 5 | zzz |

*SELECT * FROM t1 , t2;*

***ou***

*SELECT* FROM t1 cross join t2;*

| no | nom | no | valeur |
|---|---|---|---|
| 1 | a | 1 | xxx |
| 1 | a | 3 | yyy |
| 1 | a | 5 | zzz |
| 2 | b | 1 | xxx |
| 2 | b | 3 | yyy |
| 2 | b | 5 | zzz |
| 3 | c | 1 | xxx |
| 3 | c | 3 | yyy |
| 3 | c | 5 | zzz |

(9 rows)

*SELECT* FROM t1, t2 WHERE t1.no=t2.no;*

***ou***

*SELECT * FROM t1 INNER JOIN t2 on t.no=t2.no;*

| no | nom | valeur |
|---|---|---|
| 1 | a | xxx |
| 3 | c | yyy |

# Tables jointes (2/2)

***SELECT * FROM t1 LEFT JOIN t2 ON t1.no = t2.no ;***

| no | nom | no | valeur |
|---|---|---|---|
| 1 | a | 1 | xxx |
| 2 | b | | |
| 3 | c | 3 | yyy |

(3 rows)

***SELECT * FROM t1 RIGHT JOIN t2 ON t1.no = t2.no;***

| no | nom | no | valeur |
|---|---|---|---|
| 1 | a | 1 | xxx |
| 3 | c | 3 | yyy |
| | | 5 | zzz |

(3 rows)

## Transactions (1/4)

***Une transaction*** assemble plusieurs étapes en une seule opération **tout-ou-rien**.
Si un échec survient , alors aucune des étapes n'affecte la base de données.

Une transaction est déclarée en entourant les commandes SQL par les clauses ***BEGIN*** et ***COMMIT***.

# Langage SQL

## Transactions (2/4)

Prenons comme exemple une transaction bancaire :

```
BEGIN;
    UPDATE comptes SET balance = balance - 100.00 WHERE nom = 'Alice';
    UPDATE branches SET balance = balance - 100.00
    WHERE nom = (SELECT nom_branche FROM comptes WHERE nom = 'Alice');
    UPDATE comptes SET balance = balance + 100.00  WHERE nom = 'Bob';
    UPDATE branches SET balance = balance + 100.00
    WHERE nom = (SELECT nom_branche FROM comptes WHERE nom = 'Bob');
COMMIT;
```

## Transactions (3/4)

La commande **ROLLBACK** peut être utilisée à la place de **COMMIT** pour annuler la transaction.

Un groupe d'instructions entourées par **BEGIN** et **COMMIT** est parfois appelé ***bloc transactionnel***.

Il est possible d'augmenter la granularité du contrôle des instructions au sein d'une transaction en utilisant des *points de retournement* (**savepoint**).

Ceux-ci permettent d'annuler des parties de la transaction tout en validant le reste. Après avoir défini un point de retournement à l'aide de **SAVEPOINT**, les instructions exécutées depuis ce point peuvent, au besoin, être annulées avec **ROLLBACK TO**.

# Langage SQL

## Transactions (4/4)

Exemple:

```
BEGIN;
    UPDATE comptes SET balance = balance - 100.00 WHERE nom = 'Alice';
    SAVEPOINT mon_pointdesauvegarde;
    UPDATE comptes SET balance = balance + 100.00 WHERE nom = 'Bob';
    -- oups ... annulons à partir de 'mon_pointdesauvegarde'  et créditons le compte de Wally
    ROLLBACK  TO  mon_pointdesauvegarde;
    UPDATE comptes SET balance = balance + 100.00 WHERE nom = 'Wally';
COMMIT;
```

## Vues

*CREATE [ OR REPLACE ] VIEW nomVue AS requête*

**OR REPLACE** : *à utiliser si on veut remplacer une vue existante*

**A quoi servent les vues ?**

Les vues peuvent être utilisées pour différentes raisons. Elles permettent par exemple de :

- Restreindre l'accès aux données confidentielles par type d'utilisateur.
- Masquer la complexité du schéma.
- Mise à jour automatiquement des données calculées ou d' aggrégation (*sum*(), *avg*(), *max*(),...).

# Langage SQL

## Expressions conditionnelles (1/2)

CASE

```
CASE
      WHEN condition THEN résultat
      [WHEN ...]
      [ELSE résultat]
END
```

*Exemple* :

```
SELECT a  FROM test;
a
---
 1
2
3
---------------------------------------
 SELECT a,  CASE
                    WHEN a=1 THEN 'un'
                    WHEN a=2 THEN 'deux'
                   ELSE 'autres'
              END  as  description
FROM test;
a  | description
---+-------
 1  | un
2  | deux
3  | autres
```

# Langage SQL

## Expressions conditionnelles (2/2)

**COALESCE**

*COALESCE(valeur [, ...])*

La fonction COALESCE renvoie le premier de ces arguments qui n'est pas nul. Une valeur NULL est renvoyée seulement si tous les arguments sont nuls.

**NULLIF**

*NULLIF(valeur1, valeur2)*

La fonction NULLIF renvoie une valeur NULL si *valeur1* et *valeur2* sont égales : sinon, il renvoie *valeur1*.

**GREATEST et LEAST**

*GREATEST(valeur [, ...])*
*LEAST(valeur [, ...])*

Les fonctions *GREATEST* et *LEAST* sélectionnent, respectivement, la valeur la plus grande et la valeur la plus petite à partir d'une liste d'un certain nombre d'expressions

# Langage SQL

## Clauses GROUP BY et HAVING

**GROUP BY** permet de grouper les lignes d'enregistrement.
**HAVING** spécifie une condition à vérifier sur les lignes groupées.
***Exemple***:

```
SELECT ville, max(t_basse)
FROM temps
GROUP BY ville;
```

| ville | max |
|---|---|
| Hayward | 37 |
| San Francisco | 46 |

(2 rows)

```
SELECT ville, max(t_basse)
FROM temps
GROUP BY ville HAVING
max(t_basse) < 40;
```

| ville | max |
|---|---|
| Hayward | 37 |

(1 row)

# Langage SQL

## Combiner des requêtes

Les résultats de deux requêtes peuvent être combinés en utilisant les opérations d'ensemble : **union**, **intersection** et **différence**.
La syntaxe est :

***requete1 UNION [ALL] requete2***
***requete1 INTERSECT [ALL] requete2***
***requete1 EXCEPT [ALL] requete2***

➢ ***UNION*** élimine les lignes dupliquées du résultat, de la même façon que DISTINCT, sauf si UNION ALL est utilisée.

➢ ***INTERSECT*** renvoie toutes les lignes qui sont à la fois dans le résultat de *requete1* et dans le résultat de *requete2*. Les lignes dupliquées sont éliminées sauf si INTERSECT ALL est utilisé.

➢ ***EXCEPT*** renvoie toutes les lignes qui sont dans le résultat de *requete1* mais pas dans le résultat de *requete2* (*différence* entre deux requêtes). De nouveau, les lignes dupliquées sont éliminées sauf si EXCEPT ALL est utilisé.

# Langage SQL

## CREATION INDEX

L'utilisation d'un index simplifie et accélère les opérations de recherche, de tri, de jointure ou d'agrégation effectuées par le SGBD.

*CREATE INDEX nomIndex ON nomTable ( colonne1[, colonne2] [,...])*

**Exemples:**

Créer un index sur la colonne titre dans la table films :

*CREATE INDEX title_idx ON films (title);*

Pour supprimer un index créé:

*ALTER TABLE NomTable DROP INDEX nomIndex;*

# Langage SQL

## Ajouter une contrainte

Pour ajouter une contrainte, la syntaxe de contrainte de table est utilisée.

Exemples :

*ALTER TABLE produits ADD CONSTRAINT nom_contrainte_unique UNIQUE (no_produit);*

*ALTER TABLE produits ADD CONSTRAINT nom_contrainte_foreign_key FOREIGN KEY (id_groupe_produits) REFERENCES groupe_produits;*

*Pour enlever respectivement les contraintes:*

*ALTER TABLE produits DROP INDEX nom_contrainte_unique ;*

*ALTER TABLE produits DROP FOREIGN KEY nom_contrainte_foreign_key ;*

Pour ajouter une contrainte ***NOT NULL***, qui ne peut pas être écrite sous forme d'une contrainte de table, la syntaxe suivante est utilisée :

*ALTER TABLE produits MODIFY no_produit varcha(20) not null;*

*Pour enlever **NOT NULL** :*

*ALTER TABLE produits MODIFY no_produit varcha(20) null;*

La contrainte étant immédiatement vérifiée, les données de la table doivent satisfaire la contrainte avant qu'elle ne soit ajoutée.

# Langage SQL

## Supprimer une contrainte

La suppression se fait par son nom si elle a été explicitement nommée. Dans le cas contraire, le système engendre et attribue un nom qu'il faut découvrir à partir de la commande **show create table nom_table**

La commande est :

***ALTER TABLE nom_table DROP FOREIGN KEY nom_contrainte;***

Exemple:

***ALTER TABLE commande DROP FOREIGN KEY cmmande_fki_client_id;***

# Langage SQL

## Modifier la valeur par défaut d'une colonne

La commande de définition d'une nouvelle valeur par défaut de colonne ressemble à celle-ci :

*ALTER TABLE produits ALTER COLUMN prix SET DEFAULT 7.77;*

Pour retirer toute valeur par défaut, on écrit :

*ALTER TABLE produits ALTER COLUMN prix DROP DEFAULT;*

# Langage SQL

**Modifier le type de données d'une colonne:**

*ALTER TABLE produits MODIFY  prixUnitaire numeric(10,2);*

*ALTER TABLE client MODIFY  nom  varchar(150) not null;*

**Renommer une colonne:**

*ALTER TABLE Client CHANGE COLUMN nom  nomClient varchar(100);*

**Renommer une table:**

*ALTER TABLE produits RENAME TO article;*

# Langage SQL

## Afficher la requête de création d'une table:

*SHOW CREATE TABLE nom_table*

## Afficher les contraintes à une table:

*SELECT * FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS WHERE TABLE_NAME = nom_table*

# Plan de formation

1. MCD et MLD
2. Base de données Mysql
3. Langage SQL
4. Sauvegarde et Restauration
5. Gérer les utilisateurs et leurs droits

# Sauvegardes et restaurations

## Sauvegardes – mysqldump

Le principe est de produire un fichier texte de commandes SQL appelé «fichier dump », à partir de la commande externe mysqldump.

L'usage basique est :

mysqldump –u nomUtilisateur [options] *base_de_donnees > fichier_de_sortie*

*Options: on a plusieurs options mais on donne ici 2 options les plus utilisées:*

*--all-databases : sélectionne toutes les bases de données existantets*

*--databases bd1 bd2 bd3 : sélectionne les base de données db1 db2 db3*

*-d : permet de ne tenir compte que les structures des tables de la base de données*

Exemple :

Sauvegarder la base de données **maBoutique** sous le nom dans le répertoire **C:\TEST :**

**Données + structure:**

*mysqldump -u root maBoutique > c:\TEST\maBoutique_dump.sql*

**Structure seulement**:

*mysqldump -u root –d maBoutique > c:\TEST\maBoutique_structure.sql*

# Sauvegardes et restaurations

## Sauvegardes – mysqldump

Quelques exemples avec les options :

Pour sauvegarder toutes les bases de données on utilise la commande suivante :

```
mysqldump --user=mon_user --password=mon_password --all-databases > fichier_destination.sql
```

Pour sauvegarder une base de données précise :

```
mysqldump --user=mon_user --password=mon_password --databases nom_de_la_base >
fichier_destination.sql
```

Pour sauvegarder plusieurs bases de données :

```
mysqldump --user=mon_user --password=mon_password --databases nom_de_la_base_1 nom_de_la_base_2 >
fichier_destination.sql
```

Pour sauvegarder une table précise :

```
mysqldump --user=mon_user --password=mon_password --databases nom_de_la_base --tables
nom_de_la_table > fichier_destination.sql
```

Pour sauvegarder plusieurs tables :

```
mysqldump --user=mon_user --password=mon_password --databases nom_de_la_base --tables
nom_de_la_table_1 nom_de_la_table_2 > fichier_destination.sql
```

# Sauvegardes et restaurations

## Sauvegarde automatique de BDD en Windows

Pour se faire, mettre dans une commande batch (.bat) les différentes lignes de commande ci-dessous (par exemple c:\sauvegardeSql.bat):

```
SET JOUR=%date:~-10,2%
SET ANNEE=%date:~-4%
SET MOIS=%date:~-7,2%
SET HEURE=%time:~0,2%
SET MINUTE=%time:~3,2%
SET SECOND=%time:~-5,2%
SET REPERTOIR=C:\SauvegardeMysql
SET
FICHIER=%REPERTOIR%\Sauvegarde_du_%JOUR%_%MOIS%_%ANNEE%_à_%HEURE%h_%MINUTE%mn
.sql
IF NOT exist  %REPERTOIR%  md  %REPERTOIR%
mysqldump –u nomUtilisateur –ppassword maBasedeDonnee  > ”%FICHIER% ”
```

**Remarque**:

Les guillemets sur la variable FICHIER nous permet d'accepter les espace dans le nom du fichier de sauvegarde.

Utiliser le Planificateur de taches Windows pour l'automatisation.

Un exemple de fichier resultat: **Sauvegarde_du_03_01_2014_A_16h_27mn.sql**

# Sauvegardes et restaurations

## Sauvegarde automatique de BDD en Linux

Pour se faire, mettre dans une commande batch (.bat) les différentes lignes de commande ci-dessous (exemple ~/ SauvegardePostgreSql /sauvegardeSql.sh):

```
#! /bin/bash
Bdd=mabasededonnee
hm=$(date +%X)
h=${hm:0:2}
m=${hm:3:2}
fichier=${bdd }_$(date  +%A_%d-%m-%Y)_${h}_${m}mn.sql
Mysqldump –u root  -ppassword   mabasededonnee  >  $fichier
```

**Rendre le fichier bash en fichier executable:**

```
chmod  +x  ~/ SauvegardeSql /sauvegardeSql.sh
```

**Remarque**:

Utiliser le Planificateur de taches **gnome-schedule** pour l'automatisation.

Un exemple de fichier resultat: **mabasededonnee_Lundi_16-06-2014_12h-19mn.sql**

# Sauvegardes et restaurations

## Restaurations

Les fichiers texte créés par **mysqldump** peuvent être lus par le programme **mysql**. La syntaxe générale d'une commande de restauration est :

*Mysql –u nomUtilisateur base_de_donnees < fichier_d_entree*

où *fichier_d_entree est le fichier en sortie de la commande mysqldump. La base de données base_de_donnees n'est* pas créée par cette commande. Elle doit être créée avant d'exécuter mysql.

Exemple:

*mysql –u root maboutique1 < c:\TEST\maboutiquedump.sql*

*(maboutique1 est déjà créée avant le lancement de la commande)*

# Plan de formation

1. MCD et MLD
2. Base de données Mysql
3. Langage SQL
4. Sauvegarde et Restauration
5. Gérer les utilisateurs et leurs droits

# Gestion des utilisateurs

## Introduction (1/3)

Il existe deux phases lors d'un contrôle d'accès :

- la connexion au serveur;
- la vérification des privilèges.

L'ensemble des informations relatives à la gestion des utilisateurs est stocké dans une base de données nommée ***mysql***.

.

# Gestion des utilisateurs

## Introduction (2/3)

La base de données ***mysql*** est l'une des bases créées automatiquement lors de l'installation de MySQL. Nous pouvons y ajouter, modifier et supprimer des données, de deux manières différentes :

- en modifiant directement le contenu des tables avec les ordres « classiques » (Insert, Update, Delete).

  ***Méthode très délicate, déconseillée.***

- en utilisant les ordres **DCL** (Data Control Language) de MySQL, comme : Create User, Set Password, Grant, Revoke. ***C'est la méthode conseillée.***

# Gestion des utilisateurs

## Introduction (3/3)

Pour vous connecter à un serveur MySQL, vous devez disposer d'un ***nom d'utilisateur*** et du ***mot de passe*** associé.

***Syntaxe***:
mysql [-h nom_d_hote] [-u nom_d_utilisateur] [-pvotre_mot_de_passe]

La liste des utilisateurs, ainsi que leur éventuel mot de passe, correspond au contenu de la table ***user*** de la base ***mysql***.

# Gestion des utilisateurs

## Notion de compte utilisateur

Si la plupart des SGBD caractérisent un compte par son nom d'utilisateur (login), MySQL prend en considération un paramètre supplémentaire : le nom (ou l'adresse IP) de la machine depuis laquelle l'utilisateur essaie de se connecter (appelée « hôte »).

➔ Un compte utilisateur est donc **l'association** entre **le nom d'utilisateur** et **l'hôte de cet utilisateur**.

Ainsi, vous pouvez avoir par exemple:

- root@localhost :le super administrateur travaillant directement sur le serveur
- root@provider.com : le super administrateur travaillant chez lui, avec sa connexion Internet

# Gestion des utilisateurs

## Les droits d'accès dans MySQL

La base de donnée ***mysql*** est:

- créée automatiquement à l'installation.
- la base qui gère les utilisateurs et leurs privilèges d'accès sur l'ensemble des bases de données, y compris **mysql** elle-même.

«***root***» est l'utilisateur privilégié qui est en charge d'administrer la base ***mysql***. C'est donc le seul initialement qui pourra ajouter des utilisateurs et modifier leurs droits.

Comme toutes les autres bases, ***mysql*** constituée de plusieurs tables.
Nous allons décrire les 3 plus importantes à savoir:

- Table ***« User »***
- Table ***« Db »***
- Table ***« host »***

# Gestion des utilisateurs

## La table « user » (1/2)

La table « ***user*** » recense tous les utilisateurs. Elle contient les privilèges de chacun d'entre eux . En pratique, ***les privilèges les plus importants sont réservés à un administrateur de la base***.

# Gestion des utilisateurs

## La table « user » (2/2)

| Attribut | Valeur | Fonction |
|---|---|---|
| Host | host | Serveur sur lequel mysql tourne (en principe localhost) |
| User | user | Login mysql de l'utilisateur |
| Password | pass | Mot de passe mysql (crypté) de l'utilisateur |
| Select_priv | Y/N | Sélection |
| Insert_priv | Y/N | Insertion |
| Update_priv | Y/N | Modification (de valeur) |
| Delete_priv | Y/N | Effacement |
| Index_priv | Y/N | Indexation |
| Alter_priv | Y/N | Modification (de table ou de champ) |
| Create_priv | Y/N | Creation |
| Drop_priv | Y/N | Suppression |
| Grant_priv | Y/N | Permission |
| Reload_priv | Y/N | Relancer mysql |
| Shutdown_priv | Y/N | Arreter mysql |
| Process_priv | Y/N | Processus |
| File_priv | Y/N | Lecture et écriture de fichiers (import/export) |

# Gestion des utilisateurs

## La table « db »

La table « ***db*** » permet de lier un utilisateur à une ou plusieurs base(s).

C'est dans cette table que l'on peut donner des privilèges à un utilisateur

| Attribut | Valeur | Fonction |
|---|---|---|
| Host | Host | Serveur sur lequel mysql tourne (en principe localhost) |
| Db | db | Base de données |
| User | user | Utilisateur |
| Select_priv | Y/N | Sélection |
| Insert_priv | Y/N | Insertion |
| Update_priv | Y/N | Modification (de valeur) |
| Delete_priv | Y/N | Effacement |
| Index_priv | Y/N | Indexation |
| Alter_priv | Y/N | Modification (de table ou de champ) |
| Create_priv | Y/N | Creation |
| Drop_priv | Y/N | Suppression |
| Grant_priv | Y/N | Permission |

# Gestion des utilisateurs

## La table « host »

La table « **host** » permet de lier un hôte à une base de données.

Cette table permet de gérer les privilèges non pas par utilisateur mais par machine.

| Attribut | Valeur | Fonction |
|---|---|---|
| Host | host | Serveur sur lequel mysql tourne (en principe localhost) |
| Db | db | Base de données |
| Select_priv | Y/N | Sélection |
| Insert_priv | Y/N | Insertion |
| Update_priv | Y/N | Modification (de valeur) |
| Delete_priv | Y/N | Effacement |
| Index_priv | Y/N | Indexation |
| Alter_priv | Y/N | Modification (de table ou de champ) |
| Create_priv | Y/N | Creation |
| Drop_priv | Y/N | Suppression |
| Grant_priv | Y/N | Permission |

# Gestion des utilisateurs

## Gestion des droits d'accès: Manipulation des tables de la base Mysql (1/3)

Donner des droits d'accès aux utilisateurs peut être fait:

- en utilisant directement des requêtes ***INSERT***
- puis en demandant au serveur de recharger les tables de droits en utilisant la commande ***FLUSH PRIVILEGES***

# Gestion des utilisateurs

## Gestion des droits d'accès: Manipulation des tables de la base Mysql (2/3)

***Exemple*** **:** ajouter un utilisateur « ***toto*** » et lui donner accès uniquement à la base « ***donnees_de_toto*** »

**INSERT INTO user** (*host,user,password*)
**VALUES**('localhost','toto',PASSWORD('toto_pass'));

**INSERT INTO** db
(*host,user,db,Select_priv,Insert_priv,Update_priv,Delete_priv,Create_priv,Index_priv,Alter_priv*)
**VALUES** ('localhost','toto','donnees_de_toto','Y','Y','Y','Y','Y','Y','Y');

**FLUSH PRIVILEGES;**

# Gestion des utilisateurs

## Gestion des droits d'accès: Manipulation des tables de la base Mysql (3/3)

***N'oubliez pas que le mot de passe doit être encrypté!!!!***

Il est généré à l'aide de la fonction ***PASSWORD***() de MySQL.

Pour le changer :

**UPDATE** user **SET** password=password('nouveau_mot_de_passe')
WHERE user='toto';

## Afficher tous les utilisateurs du serveur mysql

select user,host from mysql.user;

# Gestion des utilisateurs

## Gestion des droits d'accès : Utilisation les ordres DCL (Data Control Language)

Donner des droits d'accès aux utilisateurs en utilisant les ordre DCL se déroule en deux étapes :

- Création de l'utilisateur;
- Assignation des privilèges

# Gestion des utilisateurs

## Gestion des droits d'accès : Utilisation les ordres DCL (Data Control Language)

### *Création des utilisateurs*

La commande ***CREATE USER*** crée un nouveau compte MySQL.

Pour chaque compte, CREATE USER crée un nouvel enregistrement dans la table mysql.user, sans aucun droit. Une erreur survient si le compte existe déjà. Le compte peut recevoir un mot de passe avec la clause optionnelle IDENTIFIED BY.

**Syntaxe de CREATE USER :**

**CREATE USER** ***user*** [IDENTIFIED BY [PASSWORD] '***password***'] [, ***user*** [IDENTIFIED BY [PASSWORD] '***password***']] ...

# Gestion des utilisateurs

## Gestion des droits d'accès : Utilisation les ordres DCL (Data Control Language)

***Création des utilisateurs exemples :***

CREATE USER 'moi'@'localhost' IDENTIFIED BY 'oim' : création d'un utilisateur 'moi' de mot de passe 'oim' sur le serveur lui-même

Create User moi@monordi ; L'ordinateur appelé monordi

Create User moi@192.168.1.123 ; L'ordinateur dont l'IP est 192.168.1.123

Create User moi@'l92.l68.%' ; N'importe quel ordinateur dont l'IP est de la classe 192.168

Create User moi@'%.microapp.com' ; N'importe quel ordinateur du domaine microapp.com

Create User moi@'%' ; N'importe quel ordinateur

# Gestion des utilisateurs

## Gestion des droits d'accès : Utilisation les ordres DCL (Data Control Language)

***Ajout de droits (1/2)***

Les droits donnés à un utilisatehelp ur sont définis directement selon qu'on leur autorise un accès large ou restreint aux bases, aux tables, et même aux colonnes d'une table. Ces droits pourront être de  insertion,consultation, destruction, ...

**Syntaxe de GRANT**

**GRANT** priv_type [(column_list)] [, priv_type [(column_list)]] ...

**ON** {tbl_name | * | *.* | db_name.*}

**TO** user 1[, user2] ...

[**WITH**  [GRANT OPTION | MAX_QUERIES_PER_HOUR count | MAX_UPDATES_PER_HOUR count | MAX_CONNECTIONS_PER_HOUR count]]

Les options MAX_QUERIES_PER_HOUR, MAX_UPDATES_PER_HOUR , MAX_CONNECTIONS_PER_HOUR  permettent de définir respectivement le nombre de requêtes maximal, le nombre de mises à jour maximal et le nombre de connexions maximal (le tout en nombre par heure).

# Gestion des utilisateurs

## Gestion des droits d'accès : Utilisation les ordres DCL (Data Control Language)

***Ajout de droits (2/3) : Liste non exhaustives des privilèges pouvant être accorder aux utilisateurs***

- ***ALL [PRIVILEGES*] :** Autorise tous les privilèges simple
- ***ALTER*** : Autorise l'usage de ALTER TABLE
- ***CREATE*** : Autorise l'usage de CREATE TABLE
- ***DELETE*** : Autorise l'usage de DELETE
- ***DROP*** : Autorise l'usage de DROP TABLE.
- ***INDEX*** : Autorise l'usage de of CREATE INDEX et DROP INDEX
- ***INSERT*** : Autorise l'usage de INSERT
- ***SELECT*** : Autorise l'usage de SELECT
- ***SHOW DATABASES :*** Autorise l'usage de SHOW DATABASES pour montrer toutes les bases
- ***SHUTDOWN*** : Autorise l'usage de l'arrêt par mysqladmin
- ***UPDATE*** : Autorise l'usage de UPDATE
- etc.

# Gestion des utilisateurs

## Gestion des droits d'accès : Utilisation les ordres DCL (Data Control Language)

### *Ajout de droits (3/3) : exemples*

**Exemple 2 : D**onner à l'utilisateur 'toto' tous les droits d'accès à toutes les bases à partir de n'importe quelle machine.

**GRANT** ALL PRIVILEGES ON * TO toto@'%' IDENTIFED BY 'password';

**Exemple 3 : D**onner à tout le monde les droits de consultation de la base Fournisseurs.

**GRANT** SELECT ON Fournisseurs TO PUBLIC;

**Exemple 4 :** Donner à Albert et Marcel le droit de mise à jour sur la base Fournisseurs.

**GRANT** UPDATE ON Fournisseurs TO Albert,Marcel;

**Remarque**:

***La commande GRANT permet aussi de créer un compte si ce dernier n'existe pas encore !!!***

# Gestion des utilisateurs

## Gestion des droits d'accès : Suppression des droits

Pour enlever les droits sur les utilisateurs on utilise la commande REVOKE

Elle est sensiblement la même que celle de l'ajout

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

.

**Exemple :** supprimer à Albert le droit de mise à jour sur la base Fournisseurs.

**REVOKE** UPDATE ON Fournisseurs FROM Albert@localhost;

# Gestion des utilisateurs

**Afficher les droits à un utilisteur:**

*Show grants for nom_utilisateur@'designation_host'*

**Afficher tous les droits de l'utilisateur en cours:**

*Show grants*

**Afficher l'utilisateur en cours:**

*Select current_user*

# Gestion des utilisateurs

## Gestion des droits d'accès : Changement de mot de passe

Pour assigner un mot de passe à un compte , on utilise la commande SET PASSWORD :

mysql> **SET PASSWORD FOR 'jeffrey'@'%' = PASSWORD('biscuit');**

Seuls les utilisateurs root ayant des accès en écriture à la base mysql peuvent changer les mots de passe des autres utilisateurs.

Pour modifier le mot de passe de l'utilisateur courant, on peut omettre la clause FOR :

mysql> **SET PASSWORD = PASSWORD('biscuit');**

# Sources

- Livres :
  - ***Audit et optimisation MySQL 5: Bonnes pratiques pour l'administrateur*** Par Pascal Borghino, Olivier Dasini, Arnaud Gadal
  - ***MySQL*** Par Paul DuBois

- Sites :
  - http:// www.developpez.com
  - http://dev.mysql.com/
  - http://sql.sh/