MySQL : Liste des instructions principales du cours

Ceci est un résumé des instructions principales utilisées au cours. Plus de détails et d'exemples seront vus dans le cadre du cours et dans les références mentionnées.

CREATE

Crée une base de données ou une table.

CREATE DATABASE

```
CREATE DATABASE nom_base_de_données
```

```
CREATE DATABASE Test_cours
```

CREATE TABLE

```
CREATE TABLE nom_table (champs : définition) [options de la table]
```

```
CREATE TABLE test_cours.articles (
   `Art_ID` INT NOT NULL AUTO_INCREMENT PRIMARY KEY COMMENT 'ID Unique
Article',
   `Designation` VARCHAR( 25 ) NOT NULL COMMENT 'Libelle de l''Article',
   `P_vente` FLOAT( 6.2 ) NOT NULL COMMENT 'Prix de Vente',
   ) ENGINE = InnoDB COMMENT = 'Articles '
```

DROP

Supprime une base de données, une table ou un index. A utiliser avec prudence

DROP TABLE

Supprime une ou plusieurs tables. Toutes les données et la structure de la table sont perdues.

```
DROP TABLE nom_table
```

DROP INDEX

Supprime un ou plusieurs index.

```
DROP INDEX nom_index ON nom_table
```

DROP DATABASE

Supprime une base de données complète : toute les tables contenues sont perdues.

```
DROP DATABASE nom_base_données
```

DELETE

Supprime des enregistrements (lignes) d'une table.

```
DELETE FROM nom_table WHERE condition
```

Sans la clause "Where" : tous les enregistrements sont effacés.

```
DELETE FROM categories WHERE type_categ = 150
```

ALTER DATABASE

Permet de modifier les caractéristiques générales d'une base de données.

```
ALTER DATABASE nom_base_donnees
```

ALTER TABLE

Permet de changer la structure d'une table existante.
Par exemple, ajouter ou supprimer des colonnes, des index, changer le type de colonnes existantes, renommer des colonnes, …

```
ALTER TABLE nom_table specifications

Avec specifications :
      ADD colonne_definiton
      ADD INDEX nom_index, type_index
      ADD PRIMARY KEY type_index
      ADD FOREIGN KEY nom_index, référence_définition
      DROP colonne
      DROP INDEX nom_index
      DROP PRIMARY KEY
      DROP FOREIGN KEY nom_clé_étrangère
      RENAME nouveau_nom_table

ALTER TABLE voitures
      ADD FOREIGN KEY (Typ_Categ)
      REFERENCES test_cours.categories (Type_Categ)
      ON DELETE CASCADE
      ON UPDATE CASCADE
```

INSERT

Ajoute des enregistrements (lignes) à une table.

```
INSERT INTO nom_table (noms_champs) VALUES (expression)
ou
INSERT INTO nom_table SET nom_champ = expression


INSERT INTO articles
      (Art_ID , Designation, P_vente )
      VALUES
      (NULL , Veste rouge', '20.14')
```

UPDATE

Mise à jour des enregistrements (lignes) d'une table avec de nouvelles valeurs.

```
UPDATE nom_table SET nom_champ1 = expression1,…
      WHERE condition
      ORDER BY nom_champs
```

SET indique les champs à modifier et les valeurs à leur donner.
Sans la clause "Where" : tous les enregistrements sont mis à jour.

```
UPDATE personnel SET age = age + 1
   Augmente l'âge de 1 an pour tout le monde

UPDATE personnel SET age = age * 2, age = age + 1
   Calcule le double de l'âge et puis l'incrémente de 1

UPDATE categories SET type_categ = 700 WHERE type_categ = 600
   Remplace le type_categ 600 par 700

UPDATE articles SET p_vente = p_vente * 1.10
   Augmente tous les prix de vente de 10%

UPDATE articles SET p_vente = p_vente * 1.25 WHERE designation LIKE '%veste%'
   Augmente les prix de vente des vestes de 25%
```

Multi-tables :

```
UPDATE articles, tarif_mensuel SET articles.prix = tarif_mensuel.prix
   WHERE articles.id = tarif_mensuel.id
         Remplace le prix des articles par celui du tarif mensuel.
```

Remarque : en multi-tables, on ne peut pas utiliser la clause ORDER BY

Affichage de certains paramètres

```
SHOW STATUS

SHOW TABLE STATUS FROM nom_base_données LIKE 'nom_table'

SHOW INNODB STATUS
```

Donne des informations spécifiques aux tables de type INNODB et notamment une explication de la dernière erreur de clé étrangère par le serveur.

SELECT

Sélection (recherche) d'informations dans les tables.

```
SELECT [DISTINCT] nom_champs
      INTO nom_fichier
      FROM nom_tables
      WHERE conditions
      GROUP BY nom_champs [ASC],[DESC]
      HAVING conditions
      ORDER BY nom_champs [ASC],[DESC]
```

```
SELECT * FROM categories
```

```
SELECT * FROM articles WHERE Designation LIKE '%Polo%'
```

```
SELECT sum(Q_stock)FROM articles WHERE Designation LIKE '%Polo%'
```

```
SELECT * FROM voitures , categories
      WHERE voitures.typ_categ = categories.type_categ
```

***Remarque** : il n'est pas possible d'utiliser un alias de champs ou une agrégation dans une clause WHERE car la valeur du champ peut ne pas être définie lorsque le WHERE est exécuté. Dans ce cas nous utiliserons la clause HAVING.*

```
SELECT nom, prenom, MAX(salaire) AS max_sal
      FROM personnel
      HAVING max_sal > 2500
```

```
SELECT CONCAT (nom,',',prenom) AS nom_complet
      FROM personnel
      ORDER BY nom_complet
```

SELECT imbriqués :

Une sous-requête doit toujours être entre parentheses.

```
SELECT * FROM table1
   WHERE id IN (SELECT id2 FROM table2)
```

```
SELECT *
   FROM categories
   WHERE type_categ IN (SELECT typ_categ FROM voitures WHERE nom LIKE "M5")
```

```
SELECT * FROM table1
   WHERE champs1 = (SELECT MAX(champs2)FROM table2)
     Trouve toutes les valeurs de la table1 qui sont égales au maximum de la
     valeur de table2
```

SELECT NOT IN :

```
SELECT * FROM table1
   WHERE id NOT IN (SELECT id2 FROM table2)
ou
SELECT * FROM table1
   WHERE NOT (id IN (SELECT id2 FROM table2))
     Trouve toutes les valeurs de la table1 qui ne sont pas dans la table2
```