Module 3 - TD: SQL, les premiers pas avec MySQL

Les premiers pas sur SQL avec le client en ligne de commande `mysql`.

- Découvrir le programme client mysql et se familiariser avec la ligne de commande
- Découvrir les instructions SQL basiques (DDL, DML)
- Se familiariser avec la documentation officielle et savoir la lire

  En cas de doute, ou envie d'en savoir plus sur une commande consulter le manuel officiel de MySQL v8.

Se connecter, créer un utilisateur, se déconnecter

Se connecter au SGBD (ouvrir une session)

```
mysql -uroot -p
--ou
mysql --user=root --password
```

Une fois connecté.e

Tester et sortir

```
-- ceci est un commentaire
-- afficher la version de MySQL et la date courante
SELECT VERSION(), CURRENT_DATE;
help;
SHOW DATABASES;
exit; -- ou QUIT ou Ctr+d
```

Créer un nouvel utilisateur et un nouvelle base de données

> Ne pas utiliser l'utilisateur `root` si vous n'êtes pas l'administrateur de la base ou si vous êtes côté applicatif !

```
CREATE USER 'dev'@'localhost' IDENTIFIED BY 'password';
-- créer une nouvelle base de données
CREATE DATABASE mod3;
-- donner tous les droits à dev sur la base de données mod3
GRANT ALL PRIVILEGES ON mod3.* TO 'dev'@'localhost';
-- test si l'utilisateur est bien crée
SELECT User, host FROM mysql.user;
-- Supprimer l'utilisateur
```

```
DROP USER 'dev'@'localhost';-- Supprimer la base de données
DROP USER 'dev'@'localhost';
```

Sans copier/coller:

1. **Recréer** l'utilisateur `dev` en tant que `root`.
2. **Recréez** la base de données `mod1` et donnez tous les droits à `dev` sur `mod1`.
3. **Lister** toutes les bases de données en tant que `dev`. Refaites-le en tant que `root`. Que remarquez-vous ?
4. **Se connecter** en tant que `dev` **sur la base de données** `mod1`.
5. **Afficher** l'utilisateur courant. (Voir CURRENT_USER).
6. **Identifier** les mots clefs SQL et commencer à vous créer un dictionnaire (mot SQL: que fait la requête).

Exécuter un script SQL en *Batch Mode*

```
mysql -h host -u user -p < mon-script.sql
# Ecrire les résultats dans le fichier mysql.out
mysql -h host -u user -p < mon-script.sql > mysql.out
# Forcer le script à continuer à s'exécuter même s'il y a des erreurs
mysql -h host -u user -p --force < mon-script.sql
```

7. **Créer** votre premier script sql `hello_world.sql` et renseigner la requête pour afficher la version de MySQL, l'utilisateur courant et la date du jour. **Exécuter** le script en *batch mode*. Récupérer le résultat dans le fichier `mysql.out`. Inspecter le contenu du fichier de sortie.

Customiser le prompt de `mysql`

Configurer le client `mysql` via le fichier `INI` `my.cnf` (ou my.ini en fonction de la version/OS)

Sous Debian

```
sudo vim /etc/mysql/my.cnf
```

Ajouter ceci dans le fichier de configuration ~~INI [mysql] prompt=(\u@\h) [\d]*mysql*>\~

> Ici on affichera l'utilisateur, l'host et la base de données courante.

Première table SQL

8. Créer dans la base de données `mod3` la relation `Mytable`. Que remarquez-vous ?

```
CREATE TABLE Mytable;
```

9. Ajouter une colonne à la table Mytable

```
CREATE TABLE Mytable (attribut INTEGER);
```

10. À l'aide de la commande `SHOW`, **inspecter**:
    1. toutes les tables de la base `mod3`
    2. les colonnes de la table `Mytable`
11. **Supprimer** la table `Mytable` avec l'instruction `DROP`

Étoffer notre première relation

On va définir nos relations et leurs attributs. On parle de *DDL* (*Data Definition Language*), un sous-ensemble des requêtes réalisables en SQL, caractérisé par les mots `CREATE`, `ALTER`, `TRUNCATE` et `DROP`.

Exemple d'une compagnie aérienne et de pilotes embauchés par la compagnie.

> Tous les concepts utilisés dans cet atelier exploratoire seront expliqués en détail par la suite.

Voici la définition des données du problème (les types de données sont indiqués entre parenthèses):

- `Compagnie`: **id** (INTEGER), numero_rue (TINYINT), nom_rue (CHAR(40)), ville (CHAR(30)), nom (CHAR(30))
- `Pilote`: **id** (INTEGER), nom (CHAR(30)), nb_heures_vol (TINYINT), id_compagnie (INTEGER)

  La clef, ou l'identifiant de chaque enregistrement est indiqué par l'attribut en gras (ici id dans chaque relation).

  Quelle différence entre INTEGER et TINYINT ? C'est une question de mémoire allouée et donc de valeurs possibles. Un TINYINT est encodé sur 1 octet (8 bits, soit 2^8= 256 valeurs possibles). Un INTEGER est encodé sur 4 octets (32 bits soit 2^{32} = 4 294 967 296 valeurs possibles) Voir ici une comparaison des types d'entiers disponibles.

Contraintes métiers:

- Un pilote est embauché par une compagnie. Une compagnie embauche un ou plusieurs pilotes.
- Une Compagnie ne peut embaucher que des pilotes **ayant au moins 10 heures de vol** (contrainte `CHECK`)
- Un Pilote **doit être embauché par une compagnie à tout moment** (sinon il sort du système, cela nous intéresse pas dans ce cas), (`NOT NULL`)
- Le nom de la rue de l'adresse de la compagnie **doit être renseigné** (`NOT NULL`)
- La ville **par défaut** de la Compagnie est 'Paris' (`DEFAULT`)

Prérequis:

- être connecté avec notre user `dev`
- avoir créé la base de données `mod3`
- *utiliser* (se connecter à) la base `mod3`
- n'avoir aucune table dans la base de données `mod3`

12. Créer un script `ddl_compagnies_aeriennes.sql` >Vérifier que vous êtes bien *sur* la base de données (connecté). Se connecter à la base soit à la connexion (`mysql -udev -p -Dmod3`), soit dans le prompt avec `USE mod3;`. Sinon vous pouvez faire `CREATE TABLE IF NOT EXISTS mod3.mytable;`, lire la documentation à ce sujet.

13. Executer le script en mode *bash*:

```
mysql -udev -p < ddl_compagnies_aeriennes.sql
```

14. Lister toutes les tables de la base `mod3` avec l'instruction SHOW.
15. Inspecter toutes les colonnes de la table `Pilote` avec l'instruction SHOW COLUMNS FROM ou DESCRIBE.

Enregistrons des *éléments* (*tuple* ou *ligne*) dans chacune de nos deux relations.

Ici, nous ne sommes plus dans la définition de données, mais dans la *manipulation* de données ou DML (*Data manipulation Language*) caractérisée par les mots `INSERT`, `DELETE` et `UPDATE` (l'équivalent du *CRUD* en SQL)

```
-- écriture de données dans la table Compagnie avec INSERT
INSERT INTO Compagnie (id, numero_rue, nom_rue, nom) VALUES (1, 12, 'chat botté
        (du)', 'Flying Airlines');
```

On peut retrouver les éléments insérés avec l'instruction `SELECT`

```
-- Récupérer tous les éléments de la table (* pour récupérer tous les
        attributs/colonnes)
SELECT * FROM Compagnie;
-- raccourci
TABLE Compagnie;
```

Clefs et contraintes

16. **Créer** la relation `Pilote`:**id**, nom, nb_h_vol, compagnie. **Ajouter** une contrainte de clef primaire sur l'attribut id.

17. **Ajouter** la contrainte sur le nombre d'heures de vol spécifiée précédemment (au moins 10h de vol). **Insérer** des données test de Pilote (trois ou quatre) et **essayer d'insérer** un pilote avec un nombre d'heures invalide (par exemple 5).

18. **Ajouter** une contrainte de *clef étrangère* sur l'attribut compagnie de la table Pilote qui référence la table **Compagnie**. À quoi cela sert-il ?

19. Essayer de **supprimer** l'enregistrement de la compagnie 'Flying Airlines'. **Supprimer** la table Compagnie. Qu'observez-vous ? Comment pouvons-nous réaliser ces deux opérations ?

Exercice

1. À quoi sert l'instruction `ALTER TABLE` ? **Ajouter** la colonne `code_postal` CHAR(8) à la table `Compagnie` (sans la recréer).

2. Comment ajouter une contrainte de clef primaire sur une colonne d'une table ? **Écrire** la commande sur la table `Foo` et l'attribut `id`

3. **Changer** la contrainte actuelle sur le nombre d'heures de vol de la relation `Pilote`. On souhaite désormais que le nombre d'heures de vol soit compris entre 50 et 20 000. Essayer d'insérer un pilote avec 2000 heures d'heures de vol ? Que remarquez-vous ? Pourquoi ? Proposer une solution et appliquer la correction à la table pour pouvoir réaliser l'enregistrement précédent.

4. **Ajouter** une contrainte d'unicité sur l'attribut `nom` de la relation `Pilote`.

5. **Lister** toutes les contraintes qui existent sur la relation `Pilote`.

6. **Récupérer** la liste des pilotes qui ont plus de 200h de vol.

Aller plus loin (en attendant)

- Suivre le tutoriel officiel de la documentation.
- Se familiariser avec les types de données de MySQL

References

- Tutoriel du manuel officiel
- Manuel officiel en ligne de MySQL v8 (en), le manuel en ligne de MySQL v8. Très complet et très bien fait. N'hésitez pas à le consulter pour éclaircir un doute, ou approfondir un sujet.
- CREATE USER Statement (Manuel)
- Batch Mode (Manuel)
- SHOW Statement (Manuel)
- CREATE TABLE Statement (Manuel)
- Data Types (Manuel)
- Using AUTO_INCREMENT
- DELETE Statement (Manuel)