

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

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 créé
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`

Voici le contenu de notre script `ddl_compagnies_aeriennes.sql`

```
-- Création de nos relations(table)

-- base de données à utiliser (ou a préciser directement avec le client mysql
  avec l'option --database=mod3 ou -Dmod3) *
-- USE est une instruction spéciale, il n'y a pas besoin de la terminer par un
  delimitateur
-- A utiliser une fois par session (connexion)
USE mod3;

-- On supprime les tables si elles existent déjà.
DROP TABLE IF EXISTS Compagnie;
DROP TABLE IF EXISTS Pilote;

-- Création de la table Compagnie
CREATE TABLE Compagnie (
  -- définition des colonnes
  id INTEGER,
  nom CHAR(30),
  numero_rue TINYINT,
  nom_rue CHAR(40) NOT NULL,
  -- nom, type, valeur par défaut, commentaire
  ville CHAR(30) DEFAULT 'Paris' COMMENT 'Par défaut: Paris',
  -- contrainte de clef primaire (au sens de Codd)
  CONSTRAINT pk_compagnie PRIMARY KEY(id)
);

-- Création de la table Pilote
CREATE TABLE Pilote(
  id INTEGER,
  nom CHAR(30),
  nb_heures_vol TINYINT,
  id_compagnie INTEGER,
  -- contrainte de clef primaire (au sens de Codd)
  CONSTRAINT pk_pilote PRIMARY KEY(id),
);
```

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 --force
```

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`](#).

```
SHOW TABLES;  
SHOW COLUMNS FROM Pilote;  
DESCRIBE Pilote;
```

Nous reviendrons sur l'insertion de données plus en détail.

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)

```
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.

Déjà fait à la 12.

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).

Il faut modifier la table `Pilote` avec l'instruction `ALTER TABLE`

```
ALTER TABLE Pilote ADD CONSTRAINT ck_nb_heures_vol_greater_or_equal_10 CHECK  
(nb_heures_vol >= 10);
```

Au moment de l'insertion (`INSERT`) ou de la mise à jour (`UPDATE`), la contrainte sera activée. Toute requête ne respectant pas la contrainte sera alors rejetée.

```
mysql> INSERT INTO Pilote (id, nom, nb_heures_vol, id_compagnie) VALUES (1,
    'John Doe', 5, 1);
ERROR 3819 (HY000): Check constraint 'ck_nb_heures_vol_greater_or_equal_10' is
    violated.
```

Correction

On insère un élément de la relation **Pilote**

```
-- contrainte ck_nb_heures_vol invalide la requête
INSERT INTO Pilote (id, nom, nb_heures_vol, id_compagnie) VALUES (1, 'John
    Doe', 5, 1);
-- Une requête valide cette fois
INSERT INTO Pilote (id, nom, nb_heures_vol, id_compagnie) VALUES (1, 'John
    Doe', 50, 1);
INSERT INTO Pilote (id, nom, nb_heures_vol, id_compagnie) VALUES (1, 'Jane
    Dafoe', 5, 1);
INSERT INTO Pilote (id, nom, nb_heures_vol, id_compagnie) VALUES (1, 'Angela
    Clark', 444, 1);
```

Comme on viole la contrainte **ck_nb_heures_vol**, notre première requête est rejetée.

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 ?

```
ALTER TABLE Pilote ADD CONSTRAINT fk_pilote_work_for_compagnie FOREIGN
    KEY(compagnie) REFERENCES Compagnie(id);
```

Cela renforce la cohérence de la base de données et permet de mettre en place l'[intégrité référentielle](#): si un enregistrement de **Pilote** fait référence à un enregistrement de **Compagnie**, alors on doit assurer que cette compagnie existe (on ne peut pas la supprimer, sinon notre enregistrement pilote comporterait une donnée invalide).

Les contraintes de clef étrangères sont **indispensables** pour **maintenir la cohérence d'une base de données** où des associations (connectivité) entre tables existent.

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 ?

```
mysql> DELETE FROM Compagnie WHERE id=1;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key
    constraint fails (`mod3`.`Pilote`, CONSTRAINT
    `fk_pilote_work_for_compagnie` FOREIGN KEY (`id_compagnie`) REFERENCES
    `Compagnie` (`id`))
```

La contrainte de clef étrangère *a prévenu la suppression* (car des pilotes référencent cette compagnie !). Ainsi, la cohérence de la base de données est conservée. Pour réaliser cette

opération, il faudrait soit: - supprimer les pilotes qui référencent la compagnie d'abord - supprimer la contrainte (**déconseillé**) - ajouter une règle lors de la suppression d'une ligne référencée (voir **CASCADE ON DELETE**)

Exercice

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

ALTER TABLE sert à modifier la structure (les attributs) d'une table, ou des contraintes.

```
ALTER TABLE Compagnie ADD COLUMN code_postal CHAR(8);
```

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**

```
-- après création
ALTER TABLE Foo ADD CONSTRAINT PRIMARY KEY(id);
-- a la création
CREATE TABLE Foo(
    id INTEGER PRIMARY KEY
)
-- recommandée, à la fin
CREATE TABLE Foo(
    id INTEGER,
    CONSTRAINT pk_foo PRIMARY KEY(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.

Regardons la documentation de [ALTER TABLE](#), l'instruction pour modifier la structure d'une table et de ses contraintes:

```
-- d'abord on supprime la contrainte
ALTER TABLE Pilote DROP CONSTRAINT ck_nb_heures_vol_greater_or_equal_10;
-- puis on ajoute la nouvelle contrainte
ALTER TABLE Pilote ADD CONSTRAINT ck_nb_heures_vol_between_50_and_20000 CHECK
    (nb_heures_vol BETWEEN 50 AND 20000);
```

On insère un pilote avec 2000 heures de vol

```
INSERT INTO Pilote (id, nom, nb_heures_vol, id_compagnie) VALUES (4, 'Johny
Doe', 1999, 1);
```

On rencontre une erreur **ERROR 1264 (22003): Out of range value for column 'nb_heures_vol' at row 1**. En effet, la colonne **nb_heures_vol** est de type **TINYINT**, qui ne

peut prendre que 256 valeurs (de -128 à 127), 1999 est une valeur qui ne peut pas être stockée sur un octet. Il faut donc modifier le type de la colonne `nb_heures_vol`. En `SMALLINT`, encodé sur 2 octets par exemple

```
ALTER TABLE Pilote MODIFY nb_heures_vol SMALLINT;  
-- on retente l'insert  
INSERT INTO Pilote (id, nom, nb_heures_vol, id_compagnie) VALUES (4, 'Johny  
Doe', 1999, 1);  
-- ok
```

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

```
ALTER TABLE Pilote ADD CONSTRAINT un_nom UNIQUE (nom);
```

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

```
SHOW CREATE TABLE Pilote;
```

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

Il faut utiliser l'opérateur de restriction `WHERE`:

```
SELECT * FROM Pilote WHERE nb_heures_vol > 200;
```

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 \(Manuel\)](#)