24/11/2025

TD R310

Base de données

SEYDINA MOHAMED BADJI
ADJA AÏDA NDIAYE FALL RT2 GROUPE R1

Compte rendu TD R310

1 et 2. Pour commencer nous nous sommes connectés à notre environnement **Mariadb** en nous déplaçant dans le dossier **Bin** de mariadb puis exécuter la commande **mariadb -u root -p**

```
C:\wamp64\bin\mariadb\mariadb11.5.2>cd C:\wamp64\bin\mariadb\mariadb11.5.2\bin

C:\wamp64\bin\mariadb\mariadb11.5.2\bin>mariadb -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 3
Server version: 11.5.2-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

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

3. Vérification de la connexion

```
MariaDB [(none)]> Select version();
+-----------------+
| version()       |
+-----------------+
| 11.5.2-MariaDB |
+-----------------+
1 row in set (0.001 sec)
```

```
MariaDB [(none)]> Show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.013 sec)

MariaDB [(none)]>
```

3. Création de la base

1. Création de la base

```
MariaDB [(none)]> create database GestionStock character set utf8mb4 collate utf8mb4_unicode_ci;
Query OK, 1 row affected (0.006 sec)
```

La commande ci-dessus crée une **base de données** compatible **avec toutes les langues**, emojis, caractère spéciaux avec un **système de comparaison insensible à la casse**

2. Sélection de la base

Cette commande permet de définir la base de données courante pour toutes les commandes SQL suivantes.

```
MariaDB [(none)]> Use GestionStock
Database changed
MariaDB [GestionStock]>
```

3. Création de la Table Produit

```
MariaDB [GestionStock]> CREATE TABLE produit (
    ->   id INT AUTO_INCREMENT PRIMARY KEY,
    ->   nom VARCHAR(100) NOT NULL,
    ->   stock INT NOT NULL,
    ->   prix DECIMAL(10,2) NOT NULL
    -> );
Query OK, 0 rows affected (0.018 sec)

MariaDB [GestionStock]>
```

`id INT AUTO_INCREMENT PRIMARY KEY`

→ L'identifiant de chaque produit est unique et s'incrémente automatiquement.

`nom VARCHAR(100) NOT NULL`

→ Nom du produit, maximum 100 caractères, ne peut pas être vide.

`stock INT NOT NULL`

→ Quantité en stock, entier obligatoire.

`prix DECIMAL(10,2) NOT NULL`

→ Prix du produit, maximum 10 chiffres dont 2 après la virgule.

4. Insertion des données dans la table

```
MariaDB [GestionStock]> INSERT INTO produit (nom, stock, prix)
    -> VALUES
    -> ('clavier mecanique', 20, 79.90),
    -> ('Souris sans fil', 50, 29.90),
    -> ('Ecran 27 pouces', 10, 199.00);
Query OK, 3 rows affected (0.018 sec)
Enregistrements: 3  Doublons: 0  Avertissements: 0
```

5. Vérification des données entrées

```
MariaDB [GestionStock]> Select * from produit;
+----+-------------------+-------+--------+
| id | nom               | stock | prix   |
+----+-------------------+-------+--------+
|  1 | clavier mecanique |    20 |  79.90 |
|  2 | Souris sans fil   |    50 |  29.90 |
|  3 | Ecran 27 pouces   |    10 | 199.00 |
+----+-------------------+-------+--------+
3 rows in set (0.001 sec)
```

6. Ajouter et relier typeProduit à la table produit
   - Création de la table typeProduit

```
MariaDB [GestionStock]> CREATE TABLE typeProduit (
    ->     id INT AUTO_INCREMENT PRIMARY KEY,
    ->     nom VARCHAR(100) NOT NULL
    -> );
Query OK, 0 rows affected (0.010 sec)
```

`CREATE TABLE typeProduit` → crée une nouvelle table appelée `typeProduit`.

- `id INT AUTO_INCREMENT PRIMARY KEY` → crée une colonne `id` qui **s'incrémente automatiquement** à chaque nouvel enregistrement et sert de **clé primaire** .

- `nom VARCHAR(100) NOT NULL` → colonne pour le **nom du type de produit**, maximum 100 caractères, obligatoire.

But : stocker les types de produits (ex. alimentation, périphérique, etc.)

   - Ajout de la colonne typeproduit_id

```
MariaDB [GestionStock]> ALTER TABLE produit
    -> ADD COLUMN typeProduit_id INT;
Query OK, 3 rows affected (0.026 sec)
Enregistrements: 3  Doublons: 0  Avertissements: 0

MariaDB [GestionStock]>
```

`ALTER TABLE produit` → modifie la table `produit`.

`ADD COLUMN typeProduit_id INT` → ajoute une nouvelle colonne `typeProduit_id` de type entier.

But : chaque produit pourra maintenant **référencer un type de produit**.

- Ajout de la clé étrangère

```
MariaDB [GestionStock]> ALTER TABLE produit
    -> ADD CONSTRAINT fk_typeProduit
    ->     FOREIGN KEY (typeProduit_id)
    ->     REFERENCES typeProduit(id);
Query OK, 3 rows affected (0.021 sec)
Enregistrements: 3  Doublons: 0  Avertissements: 0
```

`ADD CONSTRAINT fk_typeProduit` → crée une **contrainte** appelée `fk_typeProduit`.

- `FOREIGN KEY (typeProduit_id)` → cette colonne dans `produit` devient une **clé étrangère**.
- `REFERENCES typeProduit(id)` → elle référence la colonne `id` dans la table `typeProduit`.

But : s'assurer que **chaque produit a un type existant** dans `typeProduit`.

Cela crée une relation entre Produit et typeProduit pour plus de cohérence

- Insertion des types de produits

```
MariaDB [GestionStock]> INSERT INTO typeProduit (nom)
    -> VALUES
    ->     ('alimentation'),
    ->     ('périphérique'),
    ->     ('matériel portable'),
    ->     ('switch');
Query OK, 4 rows affected (0.003 sec)
Enregistrements: 4  Doublons: 0  Avertissements: 0
```

- `INSERT INTO typeProduit (nom)` → indique que tu vas insérer des valeurs dans la colonne `nom`.
- `VALUES (...)` → liste des types de produits à ajouter.
- Chaque ligne devient un enregistrement dans la table.

Résultat : 4 types de produits créés avec des `id` automatiques (1, 2, 3, 4).

- Vérification des commandes saisies

```
MariaDB [GestionStock]> select * from typeProduit;
+----+--------------------+
| id | nom                |
+----+--------------------+
|  1 | alimentation       |
|  2 | périphérique       |
|  3 | matériel portable  |
|  4 | switch             |
+----+--------------------+
4 rows in set (0.001 sec)
```

7. Création des requêtes

```
MariaDB [GestionStock]> SELECT tp.nom AS type_produit, COUNT(p.id) AS nombre_produits
    -> FROM typeProduit tp
    -> LEFT JOIN produit p ON p.typeProduit_id = tp.id
    -> GROUP BY tp.nom;
+--------------------+-----------------+
| type_produit       | nombre_produits |
+--------------------+-----------------+
| alimentation       |               0 |
| matériel portable  |               0 |
| périphérique       |               0 |
| switch             |               0 |
+--------------------+-----------------+
4 rows in set (0.014 sec)
```

`SELECT tp.nom AS type_produit` → récupère le nom du type de produit.

`COUNT(p.id) AS nombre_produits` → compte combien de produits ont ce type.

`FROM typeProduit tp` → table principale = `typeProduit`.

`LEFT JOIN produit p ON p.typeProduit_id = tp.id` → jointure **garde tous les types**, même s'il n'y a aucun produit correspondant.

`GROUP BY tp.nom` → regroupe par type pour compter les produits par type.

```
MariaDB [GestionStock]> SELECT tp.nom AS type_produit, SUM(p.prix) AS somme_prix
    -> FROM typeProduit tp
    -> JOIN produit p ON p.typeProduit_id = tp.id
    -> GROUP BY tp.nom;
Empty set (0.003 sec)
```

`SUM(p.prix)` → additionne les prix des produits pour chaque type.

`JOIN produit p ON p.typeProduit_id = tp.id` → jointure **interne**, donc **exclut les types sans produits**.

`GROUP BY tp.nom` → regroupe par type de produit.

Résultat

Empty set

Pourquoi vide ?

Parce qu' **aucun produit n'a de `typeProduit_id` défini** → jointure interne ne trouve aucune correspondance → aucun résultat.

```
MariaDB [GestionStock]> SELECT tp.nom AS type_produit, p.nom AS produit, p.prix
    -> FROM produit p
    -> JOIN typeProduit tp ON p.typeProduit_id = tp.id
    -> WHERE p.prix = (
    ->     SELECT MAX(p2.prix)
    ->     FROM produit p2
    ->     WHERE p2.typeProduit_id = tp.id
    -> );
Empty set (0.008 sec)
```

`JOIN typeProduit tp ON p.typeProduit_id = tp.id` → joint chaque produit avec son type.

- `WHERE p.prix = (SELECT MAX(...))` → filtre pour ne garder que le produit **au prix maximum de son type**.
- Sous-requête `SELECT MAX(p2.prix) ...` calcule le prix maximum pour chaque type.

Résultat

Empty set

Pourquoi vide ?

Encore une fois, **aucun produit n'a de `typeProduit_id` défini**, donc la jointure ne retourne rien et la sous-requête ne trouve pas de prix maximum.

4. Création des utilisateurs

```
MariaDB [GestionStock]> CREATE USER 'admin_app'@'localhost' IDENTIFIED BY 'AdminApp!2025';
Query OK, 0 rows affected (0.013 sec)

MariaDB [GestionStock]>
MariaDB [GestionStock]> GRANT ALL PRIVILEGES ON GestionStock.* TO 'admin_app'@'localhost';
Query OK, 0 rows affected (0.002 sec)

MariaDB [GestionStock]>
MariaDB [GestionStock]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.002 sec)

MariaDB [GestionStock]> CREATE USER 'lecteur'@'localhost' IDENTIFIED BY 'Lecteur?2025';
Query OK, 0 rows affected (0.002 sec)

MariaDB [GestionStock]>
MariaDB [GestionStock]> GRANT SELECT ON GestionStock.* TO 'lecteur'@'localhost';
Query OK, 0 rows affected (0.002 sec)

MariaDB [GestionStock]>
MariaDB [GestionStock]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.001 sec)

MariaDB [GestionStock]> CREATE USER 'api_user'@'127.0.0.1' IDENTIFIED BY 'ApiUser#2025';
Query OK, 0 rows affected (0.007 sec)

MariaDB [GestionStock]>
MariaDB [GestionStock]> GRANT SELECT, INSERT, UPDATE ON GestionStock.* TO 'api_user'@'127.0.0.1';
Query OK, 0 rows affected (0.001 sec)

MariaDB [GestionStock]>
MariaDB [GestionStock]> FLUSH PRIVILEGES;
```

5.Test des droits

```
ERROR 1364 (HY000): Field 'stock' doesn't have a default value
MariaDB [GestionStock]> INSERT INTO produit (nom, stock) VALUES
('test_lecteur', 5);
ERROR 1364 (HY000): Field 'prix' doesn't have a default value
MariaDB [GestionStock]>
```