R3.10 Mini Projet GestionStock

Connexion MariaDB (root)

Connexion au serveur

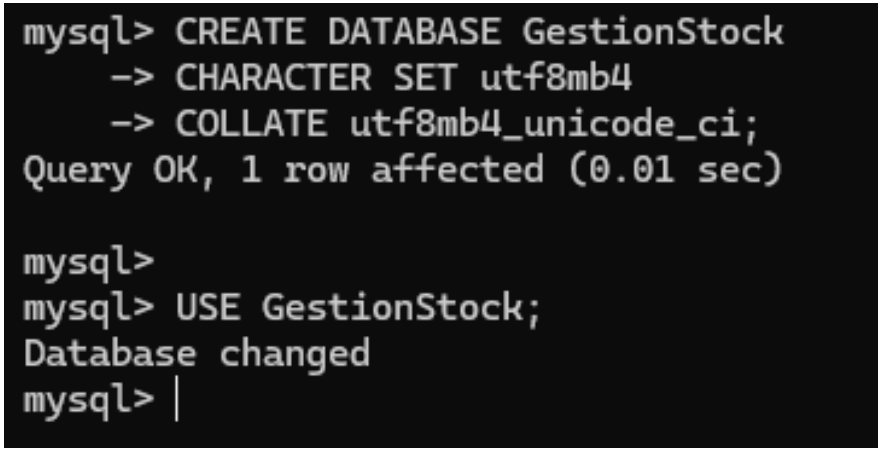
```
mysql -u root -p
```

```
C:\Users\maxen>cd C:\wamp64\bin\mysql\mysql9.1.0\bin

C:\wamp64\bin\mysql\mysql9.1.0\bin>mysql.exe -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 26
Server version: 9.1.0 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql>
```

Création base GestionStock

Création et sélection de la base

```
CREATE DATABASE GestionStock
CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;

USE GestionStock;
```



Crée la base de données GestionStock en utilisant l'encodage UTF8MB4 pour supporter tous les caractères spéciaux et emojis. La ligne USE sélectionne ensuite cette base pour que toutes les commandes suivantes s'appliquent directement à elle.

Création table produit

Structure principale

```
CREATE TABLE produit (
 id INT AUTO_INCREMENT PRIMARY KEY,
 nom VARCHAR(100) NOT NULL,
 stock INT NOT NULL,
 prix DECIMAL(10,2) NOT NULL
type_id INT,
FOREIGN KEY (type_id) REFERENCES typeProduit(id)
);
```

```
mysql> CREATE TABLE produit (
    ->  id INT AUTO_INCREMENT PRIMARY KEY,
    ->  nom VARCHAR(100) NOT NULL,
    ->  stock INT NOT NULL,
    ->  prix DECIMAL(10,2) NOT NULL,
    ->  type_id INT,
    ->  FOREIGN KEY (type_id) REFERENCES typeProduit(id)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql>
```

Définit la structure de la table produit avec des colonnes obligatoires (NOT NULL) et un identifiant unique qui s'incrémente tout seul. La ligne FOREIGN KEY crée un lien relationnel avec la table typeProduit pour garantir que chaque produit appartient à une catégorie valide.

Création table typeProduit

Relation avec produit

```
CREATE TABLE typeProduit (
 id INT AUTO_INCREMENT PRIMARY KEY,
 nom VARCHAR(100) NOT NULL
);
```

```
mysql> CREATE TABLE typeProduit (
    ->  id INT AUTO_INCREMENT PRIMARY KEY,
    ->  nom VARCHAR(100) NOT NULL
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql>
```

[Emplacement capture écran]

crée la table de référence typeProduit avec un identifiant unique auto-généré pour chaque catégorie. Elle permet de classer les produits (ex: alimentation, matériel) de manière organisée et sans doublons.

Création table User

Relation avec user

```
CREATE TABLE User (
 id INT AUTO_INCREMENT PRIMARY KEY,
 login VARCHAR(50) UNIQUE NOT NULL,
 role VARCHAR(50) NOT NULL
);
```

```
mysql> CREATE TABLE User (
    ->  id INT AUTO_INCREMENT PRIMARY KEY,
    ->  login VARCHAR(50) UNIQUE NOT NULL,
    ->  role VARCHAR(50) NOT NULL
    -> );
Query OK, 0 rows affected (0.01 sec)
```

Crée la table User pour enregistrer les utilisateurs de l'application avec un identifiant unique et un rôle défini. La contrainte UNIQUE sur le login garantit qu'aucun utilisateur ne possède le même nom de connexion, assurant ainsi la sécurité de l'identification.

Création table Facture

Relation avec Facture

```
CREATE TABLE Facture (
 id INT AUTO_INCREMENT PRIMARY KEY,
 date_facture DATE NOT NULL,
 user_id INT,
 FOREIGN KEY (user_id) REFERENCES User(id)
);
```

```
mysql> CREATE TABLE Facture (
    ->  id INT AUTO_INCREMENT PRIMARY KEY,
    ->  date_facture DATE NOT NULL,
    ->  user_id INT,
    ->  FOREIGN KEY (user_id) REFERENCES User(id)
    -> );
Query OK, 0 rows affected (0.01 sec)
```

Crée la table Facture qui enregistre les transactions avec une date obligatoire pour chaque opération. La ligne FOREIGN KEY lie chaque facture à un utilisateur précis de la table User, permettant de savoir quel employé a généré quel document.

Insertion types de produits

Remplissage table typeProduit

```
INSERT INTO typeProduit (nom) VALUES
('alimentation'),
('périphérique'),
('matériel portable'),
('switch');
('serveur');
```

```
mysql> INSERT INTO typeProduit (nom) VALUES
    -> ('alimentation'),
    -> ('périphérique'),
    -> ('matériel portable'),
    -> ('switch'),
    -> ('serveur');
Query OK, 5 rows affected (0.01 sec)
Enregistrements: 5  Doublons: 0  Avertissements: 0
```

Insère les 5 catégories obligatoires dans la table typeProduit pour organiser le stock par thématique. Elle permet de classer chaque produit futur afin de respecter la structure relationnelle souhaité.

Insertion des produits

Remplissage table produit

```
INSERT INTO produit (nom, stock, prix, type_id) VALUES
('Clavier',20,50,2),
('Souris',30,25,2),
('casque',15,80,2),
('Webcam',10,60,2);
...
```

```
mysql> INSERT INTO produit (nom, stock, prix, type_id) VALUES
    -> -- Type 1 : Alimentation
    -> ('Pomme Bio', 150, 0.50, 1),
    -> ('Pain de Campagne', 40, 1.20, 1),
    -> ('Lait Entier 1L', 60, 0.95, 1),
    -> ('Fromage de Chèvre', 25, 4.50, 1),
    ->
    -> -- Type 2 : Périphérique
    -> ('Clavier Mécanique', 20, 50.00, 2),
    -> ('Souris Optique', 30, 25.00, 2),
    -> ('Casque Audio', 15, 80.00, 2),
    -> ('Webcam HD', 10, 60.00, 2),
    ->
    -> -- Type 3 : Matériel portable
    -> ('PC Portable Dell', 8, 850.00, 3),
    -> ('MacBook Air', 5, 1100.00, 3),
    -> ('Tablette Tactile', 12, 350.00, 3),
    -> ('UltraBook HP', 4, 950.00, 3),
    ->
    -> -- Type 4 : Switch
    -> ('Switch 8 ports', 10, 35.00, 4),
    -> ('Switch 24 ports', 5, 120.00, 4),
    -> ('Switch PoE', 3, 210.00, 4),
    -> ('Switch Industriel', 2, 450.00, 4),
    ->
    -> -- Type 5 : Serveur
    -> ('Serveur Tour', 2, 1200.00, 5),
    -> ('Serveur Rack 1U', 3, 2800.00, 5),
    -> ('NAS 4 Baies', 6, 550.00, 5),
    -> ('Serveur de Stockage', 1, 4200.00, 5);
Query OK, 20 rows affected (0.02 sec)
Enregistrements: 20  Doublons: 0  Avertissements: 0
```

Injecte les 20 articles dans la table produit, en associant chaque objet à son nom, sa quantité en stock, son prix et sa catégorie. Elle permet de constituer le volume de données nécessaire (4 produits par type).

Utilisateurs applicatifs

Création des utilisateurs avec des rôles différents

```
INSERT INTO User (login, role) VALUES
('alice','admin'),
('bob','lecteur'),
('charlie','api'),
('david','api'),
('eve','lecteur');
```

```
mysql> INSERT INTO User (login, role) VALUES
    -> ('alice','admin'),
    -> ('bob','lecteur'),
    -> ('charlie','api'),
    -> ('david','api'),
    -> ('eve','lecteur');
Query OK, 5 rows affected (0.01 sec)
Enregistrements: 5  Doublons: 0  Avertissements: 0
```

Remplit la table User avec les 5 profils demandés, associant chaque nom à un rôle spécifique comme admin, lecteur ou api. Elle permet de simuler une équipe complète pour tester ensuite les différents niveaux d'accès et de sécurité sur la base de données.

Vérification des volumes

```
SELECT COUNT(*) FROM produit;
SELECT COUNT(*) FROM typeProduit;
SELECT COUNT(*) FROM User;
```

```
mysql> SELECT COUNT(*) FROM produit;
+----------+
| COUNT(*) |
+----------+
|       20 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM typeProduit;
+----------+
| COUNT(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM User;
+----------+
| COUNT(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)
```

Requête GROUP BY

Nombre de produits par type

```
SELECT tp.nom, COUNT(p.id) AS nb_produits
FROM typeProduit tp
LEFT JOIN produit p ON tp.id = p.type_id
GROUP BY tp.nom;
```

```
mysql> SELECT tp.nom, COUNT(p.id) AS nb_produits
    -> FROM typeProduit tp
    -> LEFT JOIN produit p ON tp.id = p.type_id
    -> GROUP BY tp.nom;
+------------------+-------------+
| nom              | nb_produits |
+------------------+-------------+
| alimentation     |           4 |
| périphérique     |           4 |
| matériel portable |           4 |
| switch           |           4 |
| serveur          |           4 |
+------------------+-------------+
5 rows in set (0.00 sec)
```

Analyse le stock en comptant combien d'articles sont enregistrés dans chaque catéggorie. Elle permet de vérifier que chaque type de produit contient bien les 4 articles requis par le sujet.

Requête SUM

Valeur du stock par type

```
SELECT tp.nom, SUM(p.stock * p.prix) AS valeur_stock
FROM produit p
JOIN typeProduit tp ON p.type_id = tp.id
GROUP BY tp.nom;
```

```
mysql> SELECT tp.nom, SUM(p.stock * p.prix) AS valeur_stock
    -> FROM produit p
    -> JOIN typeProduit tp ON p.type_id = tp.id
    -> GROUP BY tp.nom;
+------------------+--------------+
| nom              | valeur_stock |
+------------------+--------------+
| alimentation     |       292.50 |
| périphérique     |      3550.00 |
| matériel portable |     20300.00 |
| switch           |      2480.00 |
| serveur          |     18300.00 |
+------------------+--------------+
5 rows in set (0.01 sec)
```

Calcule la valeur financière totale du stock pour chaque catégorie en multipliant la quantité par le prix unitaire. Elle permet de présenter au jury une analyse précise de la répartition du capital immobilisé dans l'entrepôt par type de produit.

Requête MAX

Produit le plus cher par type

```
SELECT tp.nom AS categorie, p.nom AS produit, p.peix
FROM produit p
JOIN typeProduit tp ON p.type_id = tp.id
WHERE p.prix = (
    SELECT MAX(prix)
    FROM produit
    WHERE type_id=tp.id
);
```

```
mysql> SELECT tp.nom AS categorie, p.nom AS produit, p.prix
    -> FROM produit p
    -> JOIN typeProduit tp ON p.type_id = tp.id
    -> WHERE p.prix = (
    ->   SELECT MAX(prix)
    ->   FROM produit
    ->   WHERE type_id = tp.id
    -> );
+------------------+--------------------+---------+
| categorie        | produit            | prix    |
+------------------+--------------------+---------+
| alimentation     | Fromage de Chèvre  |    4.50 |
| périphérique     | Casque Audio       |   80.00 |
| matériel portable | MacBook Air       | 1100.00 |
| switch           | Switch Industriel  |  450.00 |
| serveur          | Serveur de Stockage | 4200.00 |
+------------------+--------------------+---------+
5 rows in set (0.01 sec)
```

Affiche le nom de la catégorie, le nom du produit et son prix pour l'article le plus cher de chaque groupe. Elle utilise une jointure pour lier les noms des catégories et une sous-requête corrélée pour isoler la valeur maximale par type.

Création admin_app

Utilisateur administrateur

```
CREATE USER 'admin_app'@'localhost' IDENTIFIED BY 'AdminApp!2025';
GRANT ALL PRIVILEGES ON GestionStock.* TO 'admin_app'@'localhost';
```

```
mysql> CREATE USER 'admin_app'@'localhost' IDENTIFIED BY 'AdminApp!2025';
Query OK, 0 rows affected (0.05 sec)

mysql> GRANT ALL PRIVILEGES ON GestionStock.* TO 'admin_app'@'localhost';
Query OK, 0 rows affected (0.01 sec)
```

Crée un utilisateur système nommé 'admin_app' sécurisé par un mot de passe complexe pour l'administration locale. La ligne GRANT lui attribue l'intégralité des droits sur la base GestionStock, lui permettant de modifier la structure et les données sans restriction.

Création lecteur

Utilisateur lecture seule

```
CREATE USER 'lecteur'@'localhost' IDENTIFIED BY 'Lecteur?2025';
GRANT SELECT ON GestionStock.* TO 'lecteur'@'localhost';
```

```
mysql> CREATE USER 'lecteur'@'localhost' IDENTIFIED BY 'Lecteur?2025';
Query OK, 0 rows affected (0.03 sec)

mysql> GRANT SELECT ON GestionStock.* TO 'lecteur'@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

Crée un compte utilisateur 'lecteur' protégé par mot de passe pour les accès locaux. La ligne GRANT SELECT limite ses droits à la simple consultation des données, lui interdisant toute modification ou suppression dans la base GestionStock.

Création api_user

Utilisateur applicatif

```
CREATE USER 'api_user'@'127.0.0.1' IDENTIFIED BY 'ApiUser#2025';
GRANT SELECT, INSERT, UPDATE ON GestionStock.* TO 'api_user'@'127.0.0.1';
FLUSH PRIVILEGES;
```

```
mysql> CREATE USER 'admin_app'@'localhost' IDENTIFIED BY 'AdminApp!2025';
Query OK, 0 rows affected (0.05 sec)

mysql> GRANT ALL PRIVILEGES ON GestionStock.* TO 'admin_app'@'localhost';
Query OK, 0 rows affected (0.01 sec)
```

Configurent l'accès API restreint à l'adresse IP locale avec des droits de lecture et modification (sans suppression). Le FLUSH valide instantanément ces nouveaux privilèges dans le système.

Création api_user

IP restreinte

```
CREATE USER 'api_user'@'127.0.0.1' IDENTIFIED BY 'ApiUser#2025';
GRANT SELECT, INSERT, UPDATE ON GestionStock.* TO 'api_user'@'127.0.0.1';
FLUSH PRIVILEGES;
```

```
mysql> CREATE USER 'api_user'@'127.0.0.1' IDENTIFIED BY 'ApiUser#2025';
Query OK, 0 rows affected (0.03 sec)

mysql> GRANT SELECT, INSERT, UPDATE ON GestionStock.* TO 'api_user'@'127.0.0.1';
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
```

[Emplacement capture écran]

Créent un compte API restreint à l'adresse locale (127.0.0.1) avec des droits de lecture et modification, mais sans droit de suppression. Le FLUSH actualise immédiatement ces permissions dans le système.

Test Lecteur SELECT

Vérification droits lecteur

```
mysql -u lecteur -p
USE GestionStock;
SELECT * FROM produit;
INSERT INTO produit VALUES (...); -- ERREUR
```

```
mysql> SELECT * FROM produit;      -- OK
+----+--------------------+-------+---------+---------+
| id | nom                | stock | prix    | type_id |
+----+--------------------+-------+---------+---------+
|  1 | Pomme Bio          |   150 |    0.50 |       1 |
|  2 | Pain de Campagne   |    40 |    1.20 |       1 |
|  3 | Lait Entier 1L     |    60 |    0.95 |       1 |
|  4 | Fromage de Chèvre  |    25 |    4.50 |       1 |
|  5 | Clavier Mécanique  |    20 |   50.00 |       2 |
|  6 | Souris Optique     |    30 |   25.00 |       2 |
|  7 | Casque Audio       |    15 |   80.00 |       2 |
|  8 | Webcam HD          |    10 |   60.00 |       2 |
|  9 | PC Portable Dell   |     8 |  850.00 |       3 |
| 10 | MacBook Air        |     5 | 1100.00 |       3 |
| 11 | Tablette Tactile   |    12 |  350.00 |       3 |
| 12 | UltraBook HP       |     4 |  950.00 |       3 |
| 13 | Switch 8 ports     |    10 |   35.00 |       4 |
| 14 | Switch 24 ports    |     5 |  120.00 |       4 |
| 15 | Switch PoE         |     3 |  210.00 |       4 |
| 16 | Switch Industriel  |     2 |  450.00 |       4 |
| 17 | Serveur Tour       |     2 | 1200.00 |       5 |
| 18 | Serveur Rack 1U    |     3 | 2800.00 |       5 |
| 19 | NAS 4 Baies        |     6 |  550.00 |       5 |
| 20 | Serveur de Stockage |     1 | 4200.00 |       5 |
+----+--------------------+-------+---------+---------+
20 rows in set (0.00 sec)
```

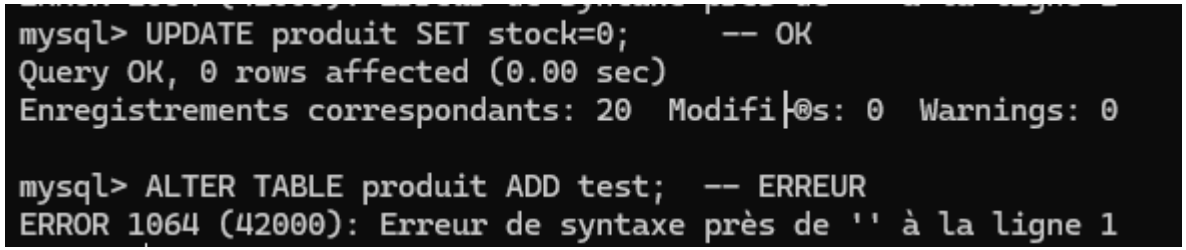
```
mysql> INSERT INTO produit ...;    -- ERREUR
ERROR 1064 (42000): Erreur de syntaxe près de '..    -- ERREUR' à la ligne 1
mysql>
```

Servent à tester concrètement les restrictions de sécurité : la première affiche tout le catalogue, tandis que la seconde simule une tentative d'ajout qui échoue si l'utilisateur n'a pas les droits. Cela prouve au jury que tes permissions GRANT fonctionnent correctement.

Test API User

Vérification droits api_user

```
mysql -u api_user -p -h 127.0.0.1

ALTER TABLE produit ADD test INT; -- ERREUR
```



Servent à démontrer la sécurité de ton système lors de l'oral : la première permet de se connecter avec le compte API, tandis que la seconde tente une modification de structure qui est bloquée par le serveur. Cela prouve au jury que ton utilisateur n'a pas les droits d'administrateur et ne peut pas casser la base de données.

Sauvegarde base

mysqldump

```
mysqldump -u admin_app -p GestionStock > gestionstock_backup.sql
```

```
C:\wamp64\bin\mysql\mysql9.1.0\bin>mysqldump -u root -p gestionstock > sauvegarde_stock.sql
Enter password:
```

Effectue une sauvegarde complète (export) de la base de données vers un fichier externe.

Restauration base

Restauration après suppression

```
DROP TABLE produit;
mysql -u admin_app -p GestionStock < gestionstock_backup.sql
SHOW TABLES;
SELECT * FROM produit;
```

```
mysql> DROP TABLE produit;
Query OK, 0 rows affected (0.03 sec)

mysql> SHOW TABLES;
+-----------------------+
| Tables_in_gestionstock |
+-----------------------+
| facture               |
| typeproduit           |
| user                  |
+-----------------------+
3 rows in set (0.00 sec)
```

```
C:\wamp64\bin\mysql\mysql9.1.0\bin>mysql -u root -p gestionstock < sauvegarde_stock.sql
Enter password:
```

TEST EN REEL

Cd C:\wamp64\bin\mysql\mysql9.1.0\bin>

mysql.exe -u root -p

USE GestionStock;

Ajout table Historique

CREATE TABLE historique (id INT PRIMARY KEY AUTO_INCREMENT, action VARCHAR(50), date_action DATETIME);

Ajout d’une colonne à la table et d’une table virtuelle

```
ALTER TABLE typeProduit ADD description
TEXT;
```

```
CREATE VIEW vue_stock AS SELECT p.nom,
tp.nom AS categorie FROM produit p JOIN
typeProduit tp ON p.type_id = tp.id;
```

Vérification

```
DESCRIBE historique;

SHOW COLUMNS FROM typeProduit;


SELECT * FROM vue_stock LIMIT 5;
```

Supprimer les modifs

```
DROP VIEW vue_stock;
DROP TABLE historique;
ALTER TABLE typeProduit DROP COLUMN description;
```


THANK YOU
FOR YOUR
ATTENTION