PROJET DE GROUPE

Gestion des Bases de Données INFO3114

GROUPE 3

Noms et prénoms :	**NI :**
- BABOU Lyes	A00235273
- SEGOUN Henock	A00225586
- TCHEFINDJIM Brice	A00224202

INTRODUCTION

Le projet de conception de la base de données a été initié pour optimiser la gestion des opérations commerciales de l'entreprise. L'objectif principal était de créer un système de base de données robuste et bien structuré permettant une gestion plus efficace des données.

I- NORMALISATION DES DONNÉES

Après avoir analysé les données existantes, nous avons identifié les besoins suivants :

- Gestion efficace des clients, produits, commandes, fournisseurs, catégories, employés et détails des commandes.
- Réduction des redondances et amélioration de l'intégrité des données.

La première étape a consisté à normaliser les données pour éviter les duplications :

- Les données ont été divisées en sept tables : Clients, Employés, Fournisseurs, Produits, Commandes, Catégories, et Détails_Commande.
- Chaque table représente une entité spécifique avec des attributs pertinents.
- Les relations entre les tables ont été définies pour maintenir l'intégrité référentielle.

Toujours dans le cadre de la normalisation des données, nous avons apporté quelques modifications spécifiques à notre projet, notamment :

- La colonne **'RequireDate'** a été supprimée, car elle contenait des dates antérieures aux dates de commende ; on ne peut prétendre passer la commande d'un article un jour et espérer le recevoir le jour passé, cela ne fait pas de sens ;
- Pour la colonne **'ShippedDate'** seules les dates les plus récentes ont été conservées pour éviter les duplications de **'OrderID'** dans la table **'Commande'** par exemple. En effet, même si la commande a été livrée à des dates différentes, cela ne fait pas de sens de les prendre toutes (ça causerait des duplicata) ou de prendre les plus anciennes dates ; en prenant les dates les plus récentes, nous sommes certains que toute les commandes ont bel et bien été livrées.

II- MODELE CONCEPTUEL DES DONNEES (MCD), MODELE LOGIQUE (MLD) ET MODELE PHYSIQUE (MPD)

a) Le Modèle conceptuel (MCD)

Le modèle conceptuel des données (MCD) a été développé à l'aide de Looping pour refléter les exigences de l'entreprise :

- **Clients** : Contient les informations sur les clients avec des relations vers les commandes.
- **Employés** : Gèrent les commandes.
- **Fournisseurs** : Fournissent les produits.
- **Produits** : Associés aux catégories et détaillés dans les commandes.
- **Commandes** : Incluent des relations avec les clients et les détails des commandes.
- **Catégories** : Classifient les produits.
- **Détails_Commande** : Contiennent les spécificités des commandes.

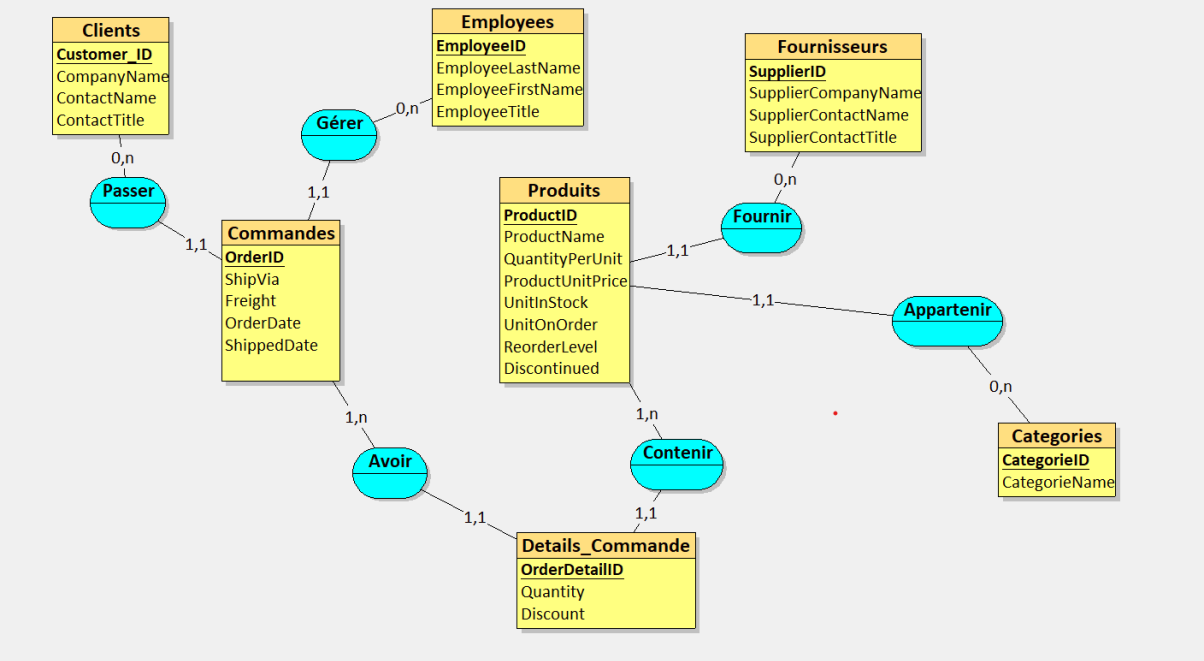


Figure 1 : Modèle Conceptuel de données

b) Le modèle relationnel ou modèle logique de données (MLD)

Le modèle Logique de données a été généré directement sur Looping à partir du MCD établi plus tôt.

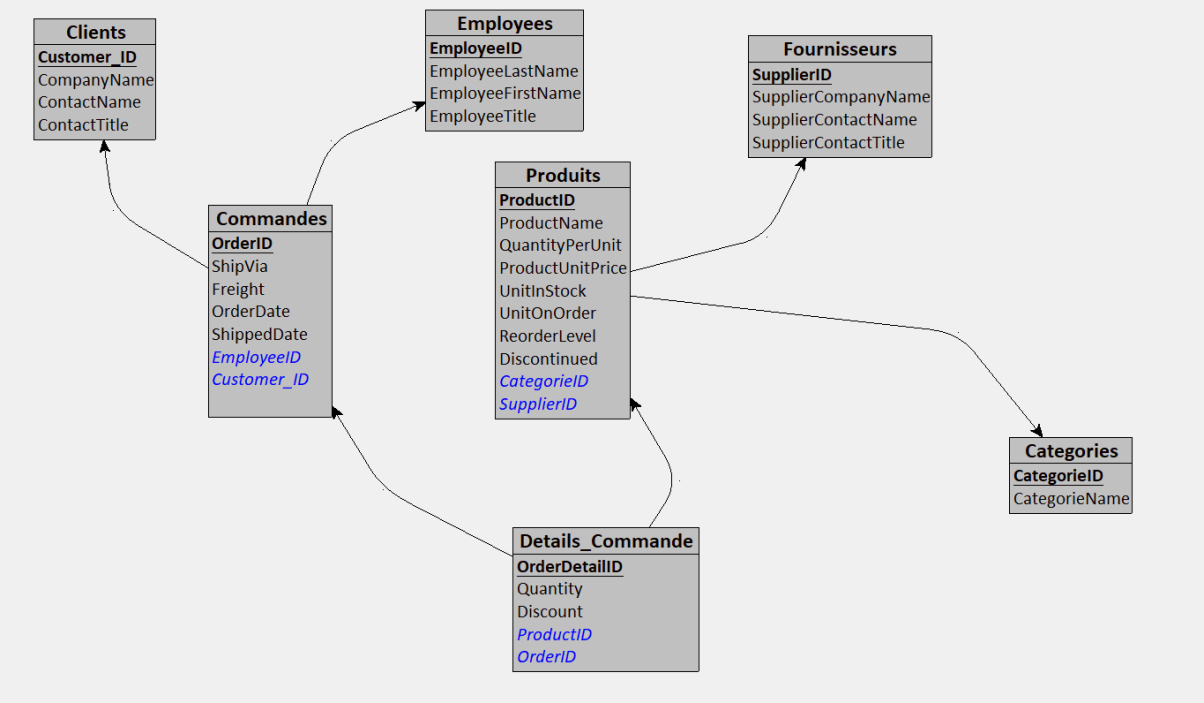


Figure 2 : Modèle Logique de données (MLD)

Clients (Customer_ID, CompanyName, ContactName, ContactTitle)

Employees (EmployeeID, EmployeeLastName, EmployeFirstName, EmployeeTitle)

Fournisseurs (SupplierID, SupplierCompanyName, SupplierContactName, SupplierContactTitle)

Categories (CatégorieID, CategoryName)

Commandes (OrderID, ShipVia, Freight, OrderDate, ShippedDate, EmployeeID, CustomerID)

Produits (productID, productName, quantityPerUnit, productUnitPrice, unitsInStock, unitsOnOrder, reorderLevel, discontinued, categoryID, supplierID)

c) Modèle physique de données

Le modèle physique également a été généré directement sur Looping à partir du MCD établi plus tôt et a été importé dans le Syst ème de Gestion des Bases de Données (SGBD) MYSQL

```
CREATE TABLE Clients(
  Customer_ID VARCHAR(50),
  CompanyName VARCHAR(100),
  ContactName VARCHAR(100),
  ContactTitle VARCHAR(100),
  PRIMARY KEY(Customer_ID)
);
```

```
CREATE TABLE Employees(
   EmployeeID INT,
   EmployeeLastName VARCHAR(50),
   EmployeeFirstName VARCHAR(50),
   EmployeeTitle VARCHAR(50),
   PRIMARY KEY(EmployeeID)
);

CREATE TABLE Fournisseurs(
   SupplierID INT,
   SupplierCompanyName VARCHAR(50),
   SupplierContactName VARCHAR(50),
   SupplierContactTitle VARCHAR(50),
   PRIMARY KEY(SupplierID)
);

CREATE TABLE Categories(
   CategorieID INT,
   CategorieName VARCHAR(50),
   PRIMARY KEY(CategorieID)
);

CREATE TABLE Commandes(
   OrderID INT,
   ShipVia INT,
   Freight REAL,
   OrderDate DATE,
   ShippedDate DATE,
   EmployeeID INT NOT NULL,
   Customer_ID VARCHAR(50) NOT NULL,
   PRIMARY KEY(OrderID),
   FOREIGN KEY(EmployeeID) REFERENCES Employees(EmployeeID),
   FOREIGN KEY(Customer_ID) REFERENCES Clients(Customer_ID)
);

CREATE TABLE Produits(
   ProductID INT,
   ProductName VARCHAR(50),
   QuantityPerUnit INT,
   ProductUnitPrice DOUBLE,
   UnitInStock INT,
   UnitOnOrder INT,
   ReorderLevel INT,
   Discontinued INT,
   CategorieID INT NOT NULL,
```

```
  SupplierID INT NOT NULL,
  PRIMARY KEY(ProductID),
  FOREIGN KEY(CategorieID) REFERENCES Categories(CategorieID),
  FOREIGN KEY(SupplierID) REFERENCES Fournisseurs(SupplierID)
);
```

```
CREATE TABLE Details_Commande(
  OrderDetailID INT,
  Quantity INT,
  Discount DOUBLE,
  ProductID INT NOT NULL,
  OrderID INT NOT NULL,
  PRIMARY KEY(OrderDetailID),
  FOREIGN KEY(ProductID) REFERENCES Produits(ProductID),
  FOREIGN KEY(OrderID) REFERENCES Commandes(OrderID)
);
```

III- CREATION DES VUES, DECLENCHEURS ET PROCEDURES STOCKEES

a) Création d'une vue

Nous avons créé une vue affichant le nombre de produits vendus ainsi que le chiffre d'affaires réalisé par chaque employé

```sql
CREATE VIEW VentesEmploye AS
SELECT
    Employees.EmployeeID,
    Employees.EmployeeLastName,
    Employees.EmployeeFirstName,
    COUNT(Details_Commande.OrderDetailID) AS NombreProduitsVendus,
    SUM(Details_Commande.Quantity * Produits.ProductUnitPrice) AS ChiffreAffaires
FROM
    Employees
JOIN
    Commandes ON Employees.EmployeeID = Commandes.EmployeeID
JOIN
    Details_Commande ON Commandes.OrderID = Details_Commande.OrderID
JOIN
    Produits ON Details_Commande.ProductID = Produits.ProductID
GROUP BY
    Employees.EmployeeID;
```

Figure 3 : Création d'une vue

b) Création d'un déclencheur

Un déclencheur (Trigger) a été créé sur la table des produits afin de mettre à jour automatiquement le stock après chaque commande

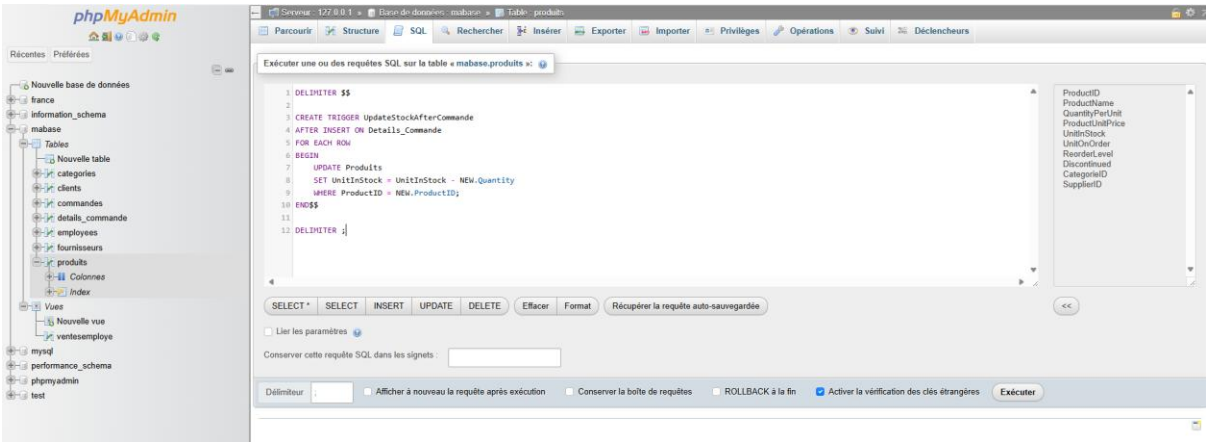


Figure 4 : Création d'un déclencheur sur la table Produit

c) Procédure stockée

Une procédure stockée a été mise en œuvre pour gérer le réapprovisionnement des produits.

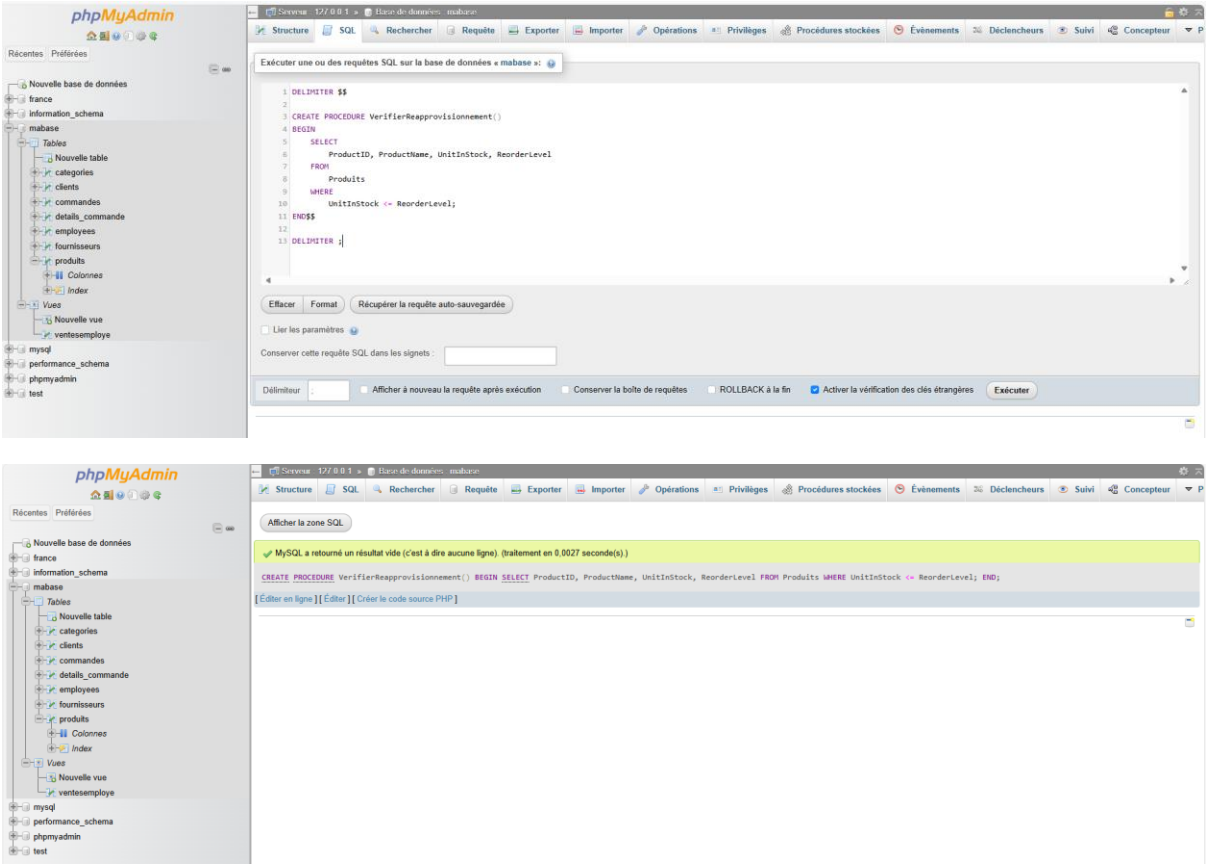


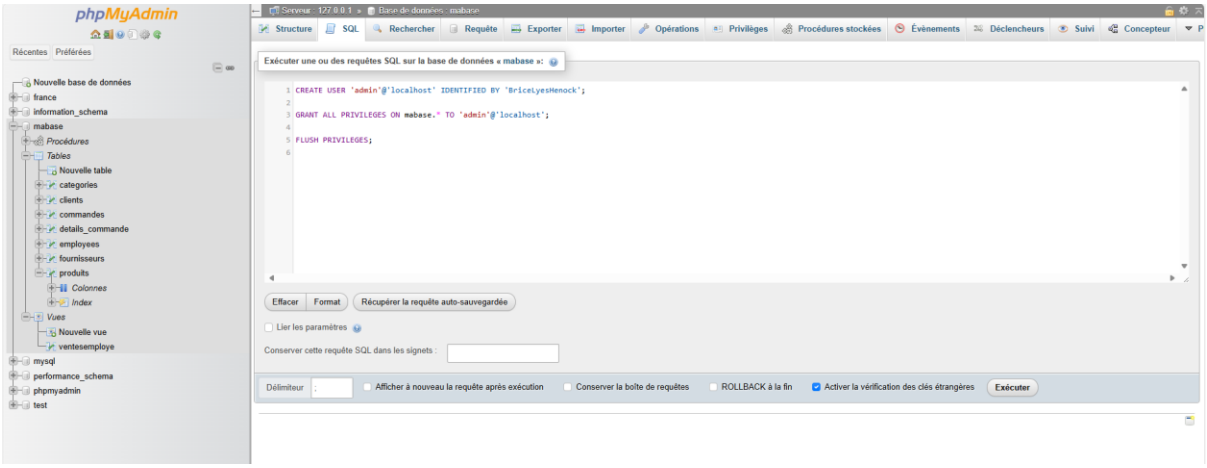
Figure 5 : Procédure stockée

IV- GESTION DES UTILISATEURS ET PRIVILEGES

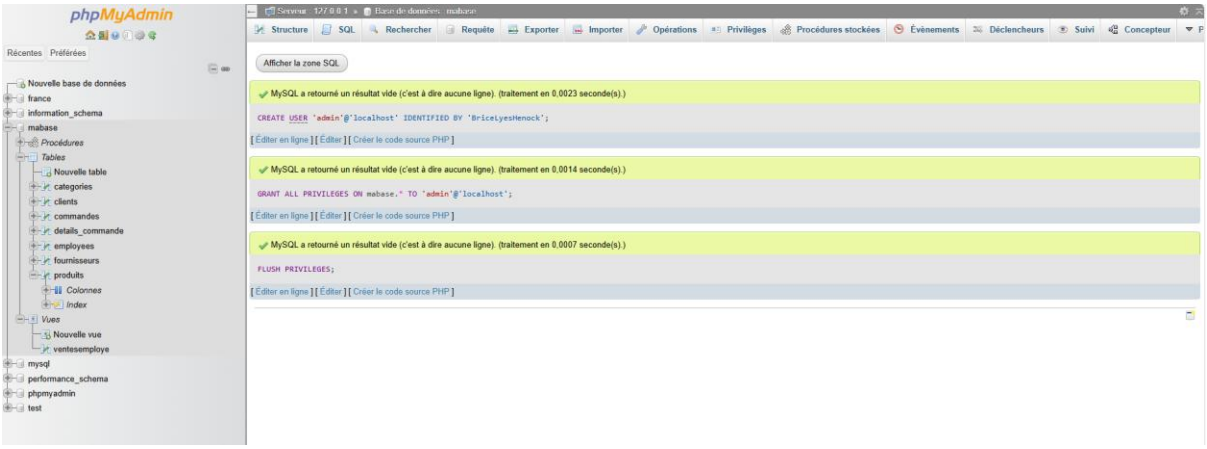
Tâche : Création de deux utilisateurs et attribution des accès à la base de données en utilisant les commandes SQL.

- **Utilisateur : "admin"** → Administrateur de base de données (**DBA**), doit disposer de tous les privilèges administratifs sur la base de données.
- **Utilisateur : "operateur"** → Analyste, doit uniquement avoir un accès en lecture.

a) <u>Création de l'utilisateur 'admin' avec tous les privillèges</u>




```
1 CREATE USER 'admin'@'localhost' IDENTIFIED BY 'BriceLyesHenock';
2
3 GRANT ALL PRIVILEGES ON mabase.* TO 'admin'@'localhost';
4
5 FLUSH PRIVILEGES;
6
```

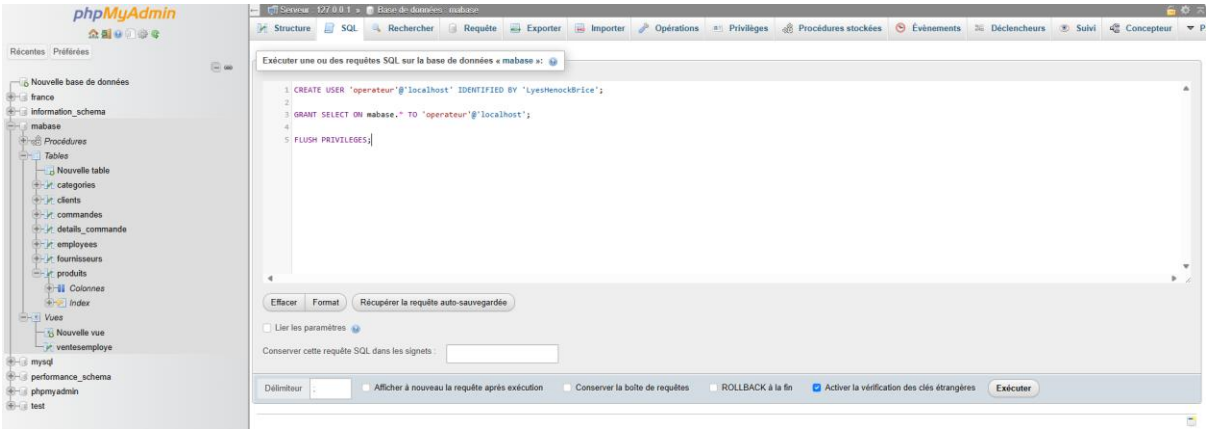


<u>Figure 6</u> : Création de l'utilisateur 'admin' avec tous les privillèges

Le mot clé **'localhost'** est utilisé par ce que on suppose que les utilisateurs se connectent depuis le même serveur.

‘BriceLyesHenock’ est le mot de passe choisi pour l’utilisateur **‘admin’**.

b) Création de l’utilisateur ‘Opérateur’ avec un accès en lecture seule




```
1 CREATE USER 'operateur'@'localhost' IDENTIFIED BY 'LyesHenockBrice';
2
3 GRANT SELECT ON mabase.* TO 'operateur'@'localhost';
4
5 FLUSH PRIVILEGES;
```

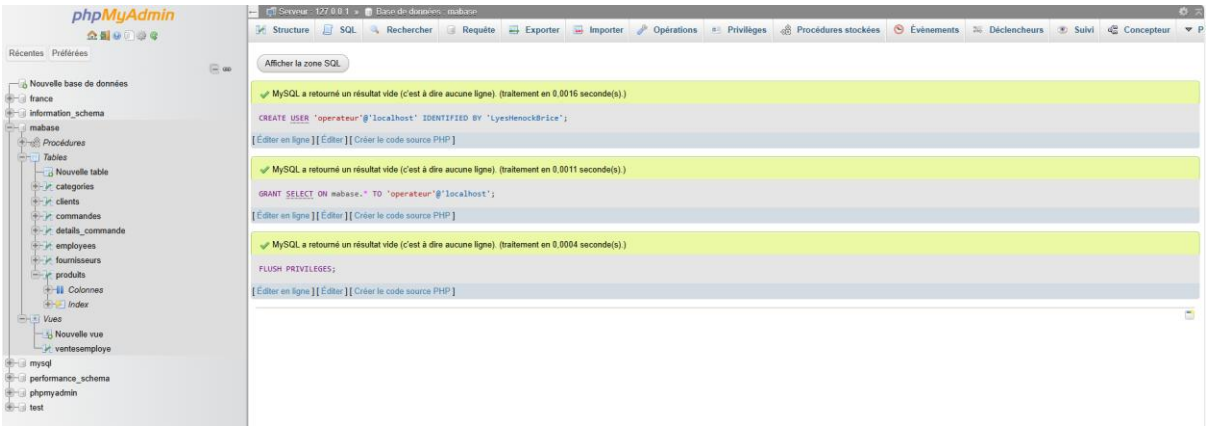


Figure 7 : Création de l’utilisateur ‘Opérateur’ avec un accès en lecture seule