**République Algérienne Démocratique et Populaire**
**Ministère de l'Enseignement Supérieur et de la Recherche Scientifique**
**Université Mouloud Mammeri de Tizi Ouzou**
Faculté des Sciences
Département Mathématiques

**Mémoire de Master**

En vue d'obtention du diplôme Master en Mathématiques Appliquées

**Spécialité : Recherche Opérationnelle**

# Conception et gestion d'une base de données relationnelle avec le langage SQL.

**Présenté par :** SIFODIL MASSIL

**Encadré par :** Dr. AMIROU AHMED

**Devant le jury d'examen composé de :**

| | | | |
|---|---|---|---|
| Mr Filali Idir | Professeur | UMMTO | Président |
| Mr Guettaf Rabah | M.C.B | UMMTO | Examinateur |
| Dr Ahmed Amirou | M.C.A | UMMTO | Promoteur |

**Année universitaire : 2024 – 2025**

# Remerciements

Je tiens à exprimer ma profonde gratitude à toutes les personnes qui ont contribué, de près ou de loin, à la réalisation de ce mémoire.

Je remercie tout d'abord mon encadreur, dr Ahmed Amirou, pour ses conseils avisés, son soutien constant, ainsi que sa disponibilité tout au long de ce travail. Sa rigueur scientifique et ses encouragements ont grandement facilité l'avancement de ce projet.

Je remercie également les membres du jury pour le temps qu'ils consacreront à l'évaluation de ce mémoire.

Mes remerciements vont aussi à mes collègues, amis et proches qui m'ont soutenu moralement et techniquement durant cette période exigeante.

Enfin, je dédie ce travail à ma famille, pour leur patience, leur amour et leur encouragement indéfectible.

# Dédicace

Avant tout, je rends grâce à Dieu Tout-Puissant, source de toute sagesse et de force. Sans Son soutien et Sa guidance, ce parcours n'aurait pas été possible. Que Sa lumière éclaire toujours mon chemin.

**À ma mère, Farida**
Je te dédie ce mémoire avec tout mon amour et ma reconnaissance. Merci pour ta patience, ton soutien inébranlable, et pour m'avoir toujours encouragé à donner le meilleur de moi-même. Ta présence est une source constante de force.

**À mon père, Farid**
Merci de m'avoir transmis des valeurs de persévérance et de travail. Ton exemple m'a toujours guidé dans mes choix et tes conseils restent gravés dans mon cœur. Ce travail est aussi le tien.

**À ma sœur, Sara**
Pour ta douceur et ta compréhension, merci d'avoir toujours été là dans les moments de doute. Ton optimisme et ta bonne humeur ont su alléger les journées difficiles.

**À mon frère, Manu**
Merci pour ta complicité et ton soutien. Ta présence m'a souvent redonné confiance et énergie pour avancer malgré les obstacles.

**À Djamila, une personne chère**
Merci d'être mon refuge, mon soutien, et ma plus belle source d'inspiration. Ta patience et ton amour m'ont porté tout au long de ce chemin. Ce mémoire, je te le dédie du fond du cœur, car tu as été à mes côtés à chaque étape.

# Table des matières

# Table des figures

# Liste des tableaux

# Introduction générale

Dans le cadre de ce mémoire, nous allons décrire les étapes clé de conception, construction et interrogatoire d'une base de données relationnelle. En un monde où l'information est un avantage stratégique pour les organisations, une bonne gestion des données est essentielle, l'information constitue un vecteur essentiel de connaissance et de communication. Vérifiable et exploitée, elle devient une donnée, qui est le pilier essentiel de tout raisonnement cherchant à résoudre une question particulière. Une base de données constitue un ensemble structuré d'informations organisée de manière à être accessible, gérée et mise à jour facilement. Les organisations l'emploient comme un outil efficace de stockage, de gestion et de récupération des données. Dans les bases de données relationnelles, les informations sont disposées selon une forme hautement structurée, c'est-à-dire qu'elles apparaissent en tables comportant un ensemble de lignes et une suite de colonnes, favorisant gestion et recherche d'informations à l'aide d'indexations et de relations établies entre les entités. Les bases sont fréquemment mises à jour selon l'évolution des besoins, à l'aide, en plus, d'ajouts, de remaniements ou suppressions de données.
En règle générale, les bases de données comportent un ensemble d'enregistrements structurés, comme les transactions commerciaux, les inventaires produits ou les profils client. Pour permettre une récupération efficace de l'information, Les données doivent être organisées de manière cohérente et efficace. Les bases de données, en revanche, fonctionnent en grande partie avec l'aide de requêtes, qui permettent l'extraction spécifique d'informations en fonction des besoins.
Un **SGBDR** est un programme qui autorise l'installation, l'usage et le maintenance des bases relationnelles. Ce modèle, développé par **Edgar F Codd** en les années **1960**, est basé sur l'organisation des données dans les tables (ou relations). Une table est formée par lignes, lesquelles sont appelées enregistrement ou tuples, et les colonnes, lesquelles sont identifiées comme les attributs. La plupart des **SGBDR** utilisent le **SQL** Language, un langage qui permet une manipulation et interrogation de données. Le SQL permet d'appliquer plusieurs opérations analogues à l'algèbre relationnelle, notamment sélection, intersection ou encore le joint. Une instruction **SQL** équivaut à une requête, qui définit une opération à effectuer par le système. Selon le type de requête, le résultat peut être affiché directement à l'écran ou utilisé pour modifier la structure ou le contenu de la base de données. Dès le début, il est important qu'il y ait un accord sur le chemin à suivre.
Dans le chapitre premier, je brosse les concepts essentiels, comme définition de donnée, définition d'une base de données et définition du langage **SQL**. J'ai commence avec un peu de généralités sur les différents types de bases de données, les SGBDR et le langage **SQL**. Le chapitre deuxième est dédié à la conception concrète d'une base de donnée, y compris définition d'une table, des champs et types de donnée liés.
Le chapitre troisième est dédié aux méthodes d'interrogation de la base de données au travers du langage **SQL**. Le chapitre quatre, concerne l'écriture et l'exécution de requêtes **SQL**.
Le chapitre cinq, l'analyse et l'existence et le fonctionnement de l'entreprise où j'ai effectué mon stage. [7][2]

# Résumé

Les bases de données relationnelles constituent un pilier fondamental des systèmes d'information modernes. Elles permettent de stocker, organiser et gérer des volumes importants de données de manière structurée. Le langage SQL (Structured Query Language) est l'outil principal utilisé pour interagir avec ces bases, offrant des commandes pour créer, modifier, interroger et manipuler les données de façon efficace et sécurisée. Le modèle relationnel repose sur des tables reliées entre elles par des clés primaires et des clés étrangères, garantissant l'intégrité et la cohérence des informations. Les principales opérations réalisées en SQL comprennent la création de bases et de tables, l'insertion de données, la modification des structures, la suppression d'enregistrements et la recherche ciblée d'informations via des requêtes adaptées. L'optimisation des bases de données vise à assurer la rapidité d'exécution des requêtes et à éviter la redondance grâce à la normalisation des tables. Une attention particulière est accordée à l'utilisation des opérateurs relationnels et logiques, des jointures, des regroupements, et des fonctions d'agrégation pour faciliter l'analyse et l'exploitation des données. Les bases de données relationnelles sont aujourd'hui largement déployées dans les entreprises industrielles et commerciales pour améliorer la gestion des stocks, le suivi des achats, la comptabilité et la prise de décision. Elles jouent également un rôle essentiel dans la vie quotidienne : elles sont au cœur des applications bancaires, des réseaux sociaux, des systèmes de réservation, des sites de commerce en ligne, et des services de santé, permettant de gérer efficacement les informations personnelles, les transactions et les services digitaux auxquels nous accédons chaque jour. De plus, des solutions modernes et automatisées émergent à travers des systèmes d'information numériques accessibles aux startups et aux petites entreprises, rendant la gestion plus flexible et performante

# Chapitre 1

# Définitions et généralités

## 1. Historique

L'histoire des bases de données remonte aux années **1960**, lorsque les entreprises ont commencé à ressentir le besoin de stocker et de gérer de grandes quantités de données de manière structurée. Avant cela, les données étaient souvent stockées sur des supports physiques comme des fiches ou des fichiers papier, ce qui rendait leur gestion fastidieuse et peu efficace. Dans les années **1960** les premiers systèmes de gestion de bases de données (**SGBD**) ont été développés. Ces systèmes étaient souvent hiérarchiques ou en réseau. IBM a été un pionnier dans ce domaine avec son système IMS (Information Management System), qui utilisait un modèle de données hiérarchique. Dans **1970**, le modèle relationnel a été introduit par **Edgar F. Codd**, un chercheur chez IBM, dans son article séminal "A Relational Model of Data for Large Shared Data Banks" (**1970**). Codd a proposé que les données soient stockées dans des tables (relations) et que les relations entre les données soient gérées par des opérations mathématiques. Ce modèle a jeté les bases de ce que nous connaissons aujourd'hui sous le nom de bases de données relationnelles.

### 1.1. Naissance de SQL

SQL (Structured Query Language) a été développé dans les années **1970** par IBM pour interagir avec les bases de données relationnelles. En1974, Donald D. Chamberlin et Raymond F. Boyce, deux chercheurs chez IBM, ont développé le langage SEQUEL (Structured English Query Language) pour interagir avec le système de base de données relationnelle System R, un projet de recherche chez IBM. SEQUEL a ensuite été renommé SQL pour des raisons de marque déposée. En **1979**, Oracle Corporation (alors appelée Relational Software, Inc.) a été la première entreprise à commercialiser un SGBD relationnel utilisant SQL. Oracle a joué un rôle clé dans la popularisation de SQL et des bases de données relationnelles. (ACM DIGITAL LIBRARY)

### 1.2. Évolution des bases de données

Au fil des années, les bases de données ont évolué pour répondre à des besoins spécifiques, notamment en termes de volume de données, de performance, et de complexité des requêtes.

— **Bases de données relationnelles :** Les bases de données relationnelles, telles que 'Oracle', 'MySQL', 'PostgreSQL', et 'Microsoft SQL Server', dominent toujours le marché. Elles sont utilisées dans une grande variété d'applications, des systèmes de gestion d'entreprise aux sites web.

— **Bases de données NoSQL :** Avec l'explosion des données non structurées et semi-structurées (comme les données JSON, XML, etc.), les bases de données 'NoSQL' (Not Only SQL) ont gagné en popularité. Ces bases de données, comme 'MongoDB', 'Cassandra', et 'Redis', sont conçues pour gérer des volumes massifs de données et offrir une grande flexibilité en termes de modèles de données.

— **Bases de données NewSQL :** Les bases de données "NewSQL", comme "Google Spanner" et "CockroachDB", combinent les avantages des bases de données relationnelles traditionnelles (comme la cohérence des données) avec la capacité de montée en charge des systèmes distribués.

### 1.3. SQL aujourd'hui

SQL reste l'un des langages de programmation les plus utilisés dans le monde, notamment en raison de sa simplicité et de sa puissance pour interagir avec les bases de données relationnelles. Il est largement utilisé dans des domaines tels que l'analyse de données, la business intelligence, et le développement d'applications web.

## 2. Définition

.

### 2.1. Définition d'une base de données

Une base de données est un ensemble structuré de données stockées électroniquement. organisé pour permettre une gestion et une mise à jour, une interrogation efficace pour répondre aux besoins informationnels d'une organisation. Elle est généralement gérée by a Système de Gestion de Bases de Données (SGBD) such as MySQL, Oracle, or PostgreSQL, which utilize le langage SQL pour manipuler les données.[11]

### 2.2. Définition d'un SGBDR

. Un SGBDR (SGBD Relationnel) est un logiciel permettant de créer, gérer et administrer des bases de données relationnelles, en s'appuyant sur les principes du modèle relationnel défini par E.F. Codd en **1970**. Il garantit l'intégrité, la sécurité et l'efficacité des données à travers :

— Le langage SQL (pour la définition, manipulation et contrôle des données)

— La gestion des schémas (tables, relations, clés primaires/étrangères). [8]

### 2.3. Définition du langage SQL

SQL (Structured Query Language) est un langage de programmation utilisé pour gérer et manipuler les bases de données relationnelles. Il permet d'effectuer des opérations telles que la création pour construire des tables et ajouter des données, la lecture pour consulter et filtrer les données, la modification pour mettre à jour des enregistrements existants, la suppression pour effacer des données ou des tables. (CRUD : Create, Read, Update, Delete). SQL est également employé pour définir la structure des bases de données et contrôler les accès aux données.[7]

## 2.4. Définition d'une table

Dans les systèmes de gestion de bases de données relationnelles, une table est une structure essentiel organisée en colonnes (attributs) et lignes (enregistrements), permettant de stocker et organiser des données de manière structurée. Chaque colonne possède un type of data spécifique et des contraintes (comme les clés primaires ou étrangères), and each line represents a unique instance of data.[5]

## 2.5. Définition d'un champ

Un champ (ou attribut) est l'unité élémentaire de stockage dans une table de base de données. Il représente une propriété spécifique d'une entité et se caractérise par :

— **un nom** (ex : nom-client, date-naissance).

— **Un type de données** (ex : VARCHAR, INT, DATE).

— **Des contraintes** (ex : NOT NULL, UNIQUE, DEFAULT).[8]

## 2.6. Définition d'une requête

Une requête est une instruction soumise à un SGBD pour interroger, manipuler ou administrer des données. Elle permet d'effectuer des opérations précises sur les tables, telles que :

— La récupération de données (SELECT).

— La modification (INSERT, UPDATE, DELETE).

— La gestion de la structure (CREATE, ALTER, DROP).[2]

## 2.7. Définition de l'entité

Une entité est un objet identifiable du monde réel, concret ou abstrait, représenté dans un système d'information par un ensemble d'attributs caractéristiques et distingué des autres objets par un identifiant unique. Dans le modèle entité-relation, elle constitue l'élément fondamental permettant de structurer conceptuellement les données avant leur implémentation physique en tables dans une base de données relationnelle.[8]

## 2.8. Définition d'une clé candidate et clé primaire

Dans une base de données relationnelle, une clé candidate est un ensemble d'attributs capable d'identifier chaque ligne de façon unique (comme le numéro de sécurité sociale ou l'email pour un client). Parmi ces clés candidates, la clé primaire est celle choisie comme identifiant principal - elle doit être unique, non nulle et stable (ex : on préfère un numéro client plutôt qu'un email qui peut changer). Cette sélection impacte directement l'efficacité des requêtes et la qualité des relations entre tables.[8]

## 2.9. Définition dune clé externe

Une clé étrangère est un champ (ou ensemble de champs) dans une table qui fait référence à la clé primaire d'une autre table. Elle garantit que les relations entre les données restent valides. Cette contrainte prévient les incohérences et facilite les requêtes entre tables associées.[8]

### 2.10. Définition des opérateurs relationnels

Les opérateurs relationnels sont des outils fondamentaux permettant de manipuler les données dans une base de données relationnelle. Ces opérateurs incluent notamment la sélection (filtrage des lignes selon une condition), la projection (extraction de colonnes spécifiques), la jointure (combinaison de tables liées), ainsi que des opérations ensemblistes comme l'union, l'intersection et la différence. Ils constituent la base théorique des requêtes SQL, fournissant un langage structuré pour interroger et transformer les données tout en garantissant leur intégrité et leur cohérence. [8]

### 2.11. Définition de l'identifiant

Un identifiant est un attribut ou un ensemble d'attributs qui permet d'identifier de manière unique chaque enregistrement d'une table. Il joue un rôle fondamental en garantissant :

1. L'unicité : Chaque valeur d'identifiant ne peut correspondre qu'à un seul enregistrement
2. L'intégrité : Il ne peut contenir de valeur nulle (NULL)
3. La stabilité : Sa valeur ne devrait pas être modifiée dans le temps

L'identifiant principal d'une table est appelé clé primaire.[1]

## 3. L'objectif d'un SGBDR

- **Stocker proprement les données**

— Ranger les informations dans des tables organisées.
— Comme un classeur bien rangé avec des ongles.

- **Retrouver vite les informations**

— Pouvoir faire des recherches en quelques secondes.
— Même dans des millions d'enregistrements.

- **Éviter les incohérences**

— Empêcher les doublons et les erreurs.
— Vérifier automatiquement que tout est correct.

- **Travailler à plusieurs en même temps**

— Permettre à plusieurs personnes de modifier les données.
— Sans que ça crée de conflits ou d'erreurs.

- **Protéger contre les accidents**

— Sauvegarder automatiquement.
— Pouvoir récupérer les données si problème.

- **Rester rapide même avec beaucoup de données**

— Bien fonctionner quand les données augmentent.
— Sans ralentir le système.

- **Sécuriser l'accès**

— Vérifier qui peut voir ou modifier les informations.
— Comme un coffre-fort avec des clés différentes.

# 4. Les logiciels les plus utilisé

Les Systèmes de Gestion de Bases de Données (SGBD) sont des outils essentiels pour stocker, organiser et manipuler des données. Voici les logiciels les plus populaires :

## 4.1. SGBD Relationnels (SGBDR) – Les Plus Courants

Ces systèmes utilisent le langage SQL et sont basés sur des tables liées entre elles.

1. Oracle Database :
   - Utilisation : Entreprises, banques, grands systèmes
   - Points forts :
     - Performances élevées
     - Sécurité renforcée
     - Supporte de très gros volumes
   - Inconvénient : Payant (coût élevé pour les licences).
2. MySQL / MariaDB :
   - Utilisation : Sites web (WordPress, Facebook, Twitter), applications
   - Points forts :
     - Open source (gratuit)
     - Rapide et facile à utiliser
     - Très répandu pour le web
   - Inconvénient : Moins adapté aux très grosses structures qu'Oracle.
3. Microsoft SQL Server :
   - Utilisation : Entreprises (surtout Windows), applications .NET
   - Points forts :
     - Intégration facile avec les outils Microsoft.
     - Bonnes performances
4. Inconvénient : Payant (mais version Express gratuite pour petits projets).
5. PostgreSQL :
   - Utilisation : Applications complexes, géodonnées, data science
   - Points forts :
     - Open source et puissant.
     - Supporte les données JSON (hybride SQL/NoSQL).
     - Meilleur pour les requêtes complexes.
   - Inconvénient : Un peu plus complexe à configurer que MySQL.

## 4.2. SGBD Cloud – Bases de Données en Ligne

De plus en plus utilisés car hébergés et gérés automatiquement.

1. Amazon RDS (Aurora, PostgreSQL, MySQL)
   - Pour : Startups et enterprises sur AWS (Amazon Web Service).
2. Google Cloud Firestore / Firebase
   - Pour : Applications mobiles et temps réel.
3. Microsoft Azure SQL Database.
   - Pour : Solutions cloud Microsoft.[18]

# Chapitre 2

# Langage SQL

Le **SQL** (Structured Query Language) est un langage standard utilisé pour gérer les bases de données relationnelles. Il a été conçu pour faciliter l'interaction avec les données. Grâce à lui, on peut créer, modifier et organiser des bases de données de manière claire et sécurisée. SQL propose des commandes simples qui permettent de définir la structure des données (comme les tables), d'ajouter ou modifier des informations, et de contrôler les accès ainsi que les opérations effectuées. Sa polyvalence et son efficacité en font un outil essentiel pour les développeurs, les administrateurs de bases de données et les analystes. SQL est largement utilisé dans la majorité des systèmes de gestion de bases de données actuels, tels que **MySQL**, **PostgreSQL**, Oracle ou **SQL Server**.

## 1. Les commandes SQL :

### 1.1. SQL CREATE DATABASE

La commande CREATE DATABASE constitue l'instruction fondamentale permettant d'initialiser une nouvelle base de données dans un système de gestion de bases de données relationnelles. S'utilise selon la syntaxe suivante :

```
    CREATE DATABASE nom-de-la-base;
```

Où "nom-de-la-base" représente l'identifiant unique attribué à la nouvelle base de données. Si une base de données porte déjà ce nom, la requête retournera une erreur. Pour éviter d'avoir cette erreur, il convient d'utiliser la requête suivante :

```
        CREATE DATABASE IF NOT EXISTS nom_de_la_base;
```

### 1.2. SQL DROPE DATABASE

La commande DROP DATABASE permet de supprimer définitivement une base de données existante, ainsi que toutes ses tables, vues, procédures et données associées. Cette opération est irréversible et doit être utilisée avec précaution. Pour supprimer la base de données " nom-de-la-base ", la requête est la suivante :

```
DROP DATABASE nom-de-la-base ;
```

## 1.3. Type de données

1. Nombres Entiers :
   - **TINYINT :** Petit nombre (-128 à 127). Ex : age TINYINT.
   - **INT :** Entier standard (ex : ID). Ex : id INT.
   - **BIGINT :** Très grands nombres. Ex : compteur BIGINT.
2. Nombres à Virgule :
   - **DECIMAL (p,s) :** Nombre précis (p=chiffres totaux, s=décimales). Ex : prix DECIMAL(10,2).
   - **FLOAT :** Approximation (pour calculs). Ex : temperature FLOAT.
3. Textes :
   - **VARCHAR(n) :** Texte de taille variable (max 255). Ex : nom VARCHAR(50).
   - **TEXT :** Texte long (descriptions). Ex : description TEXT.
4. Dates/Heures :
   - **DATE :** Format AAAA-MM-JJ. Ex : date-naissance DATE.
   - **DATETIME :** Date + heure. Ex : creation DATETIME.
5. Booléen :
   - **BOOLEAN :** TRUE/FALSE ou 1/0. Ex : actif BOOLEAN
6. Binaire :
   - **BLOB :** Stocke fichiers/images. Ex : photo BLOB
7. Enumération :
   - **ENUM :** Liste de valuers. Ex : couleur ENUM ('Rouge','Vert','Bleu').[3]

## 1.4. CREATE TABLE :

La commande CREATE TABLE est une instruction fondamentale du langage SQL qui permet de créer une nouvelle table dans une base de données relationnelle. Créer une table permet de définir les différentes colonnes ainsi que le type de données que chacune d'elles pourra contenir, comme des nombres entiers, des chaînes de caractères, des dates ou encore des valeurs binaires. La syntaxe générale pour créer une table est la suivante :

```
CREATE TABLE nom_de_la_table (
    colonne1 type_donnees [contraintes],
    colonne2 type_donnees [contraintes],
    colonne3 type_donnees [contraintes],
    ...
    contraintes_de_table
);
```

1. **Nom-de-la-table :** Identifiant unique dans la base de données
2. **Colonnes :** Chaque colonne est définie par :
   - Un nom.
   - Un type de données (INT, VARCHAR, DATE, etc.).
   - Des contraintes optionnelles (NOT NULL, UNIQUE, etc.).
3. **Contraintes de table :** Définies au niveau de la table (PRIMARY KEY, FOREIGN KEY, etc.) [10]

### 1.4.1. Exemple 1

À partir des éléments décrits ci-dessus :

— **id :** Clé primaire auto-incrémentée (identifiant unique).

— **numero-salle :** Numéro ou code de la salle (ex : "A101", "B205"), obligatoire (NOT NULL) et unique pour éviter les doublons, avec un maximum de 20 caractères aux maximum.

— **capacite :** Nombre maximum de personnes (entier, obligatoire).

— **type-salle :** Catégorie de la salle (ex : "Salle de cours", "Laboratoire"), 50 caractères aux maximum.

— **equipement :** Description libre en texte (TEXT) pour lister les équipements (ex : "Projecteur, Tableau, Prises").

— **disponible :** Booléen (TRUE/FALSE) pour indiquer si la salle est réservable (par défaut TRUE).

— **batiment :** Localisation physique, avec 50 caractères maximum.

La requête SQL de création de la table Salles est la suivante :

```
CREATE TABLE Salles (
    id INT PRIMARY KEY AUTO_INCREMENT,
    numero_salle VARCHAR(20) NOT NULL UNIQUE,
    capacite INT NOT NULL,
    type_salle VARCHAR(50) NOT NULL,
    equipement TEXT,
    disponible BOOLEAN DEFAULT TRUE,
    batiment VARCHAR(50)
);
```

### 1.4.2. Exemple 2

```
CREATE TABLE employes (
    Id INT AUTO_INCREMENT PRIMARY KEY,
    Nom VARCHAR(50),
    Prenom VARCHAR(50),
    Salaire DECIMAL(10,2)
    Departement VARCHAR (100)
);
```

— **id** : identifiant unique pour chaque employé, incrémenté automatiquement.

— **Nom et prenom** : texte jusqu'à 50 caractères.

— **Salaire** :nombre avec 2 décimales (par exemple 2500.50).

— **Departement** : texte jusqu'à 100 caractères (exemple 'informatique', 'Ressources Humaines').

## 1.5. Modification d'une table ou colonne

La modification d'une table ou d'une colonne en SQL permet d'adapter la structure d'une base de données sans perdre les données existantes.

**Contenu des Modifications Possibles :**

i. **Ajout** : Insérer une nouvelle colonne
ii. **Suppression** : Retirer une colonne existante
iii. **Changement de type** : Modifier le format des données (ex : `INT` → `VARCHAR`)
iv. **Renommage** : Donner un nouveau nom à une colonne ou une table
v. **Contraintes** : Ajouter ou supprimer des règles (ex : `PRIMARY KEY`, `NOT NULL`)

## 1.6. SQL ALTER table

ALTER TABLE est la commande qui vous permet de modifier une table existante dans votre base de données. Imaginez que vous avez créé une table, mais avec le temps, vous devez y ajouter des informations, supprimer des colonnes devenues inutiles ou corriger des types de données. C'est exactement à cela que sert ALTER TABLE. La syntaxe générale de 'ALTER TABLE' :

```
ALTER TABLE nom-table
[ACTION_A_EFFECTUER];
```

Cette commande vous permet notamment :

— D'ajouter de nouvelles colonnes pour stocker plus d'informations.
— De supprimer des colonnes qui ne servent plus.
— De modifier le type des données (par exemple changer un texte en nombre).
— De renommer des colonnes ou la table elle-même.
— D'ajouter ou supprimer des contraintes (comme des clés uniques).

L'avantage principal est que vous pouvez faire ces modifications sans perdre vos données existantes. C'est comme rénover une maison sans avoir à la détruire et la reconstruire à zéro. Cependant, sur des très grosses tables, ces opérations peuvent prendre du temps.
ALTER TABLE est donc l'outil indispensable pour faire évoluer votre base de données au fur et à mesure que vos besoins changent, tout en conservant vos précieuses informations.

— **Exemple** Imaginons qu'on a une table qui s'appelle « employés » qui désigne la matricule de l'employé 'id' et son nom, son prénom, son salaire, son département, afin de connaitre «ACTION-A-EFFECTUER » dans la syntaxe ALTER TABLE, on suit les étapes mentionnées ci-dessus :

1. **Ajouter une colonne "émail" :**

```
ALTER TABLE employés
ADD email VARCHAR(100);
```

2. **Modifier le type de "département" :**

```
    ALTER TABLE employés
    ALTER COLUMN département TYPE VARCHAR(50);
```

3. **Renommer "prénom" en "prenom-complet" :**

```
    ALTER TABLE employés
    RENAME COLUMN prénom TO prenom_complet
```

4. **Ajouter une contrainte de salaire minimum :**

```
    ALTER TABLE employés
    ADD CONSTRAINT salaire_min CHECK (salaire >=1500);
```

5. **Supprimer la colonne "département"** :

```
ALTER TABLE employés
DROP COLUMN département;
```

6. **Ajouter une colonne "date-embauche" avec valeur par défaut :**

```
ALTER TABLE employés\\
ADD date-embauche DATE DEFAULT CURRENT-DATE;
```

## 1.7. SQL DROPE TABLE

La commande **DROP TABLE** est une instruction **SQL** utilisée pour supprimer définitivement une table d'une base de données, ainsi que toutes ses données, index, contraintes et déclencheurs (triggers) associés. Pour supprimer une table " nom-de-la-table " il suffit simplement d'utiliser la syntaxe suivante :

```
DROP TABLE nom_de_la_table
```

## 1.8. SQL DELETE

La commande DELETE permet de supprimer des enregistrements spécifiques dans une table sans supprimer la table elle-même (contrairement à DROP TABLE). Pour supprimer un enregistrement dans une table " nom-de-la-table " il suffit simplement d'utiliser la syntaxe suivante :

```
DELETE FROM nom_table WHERE condition;
```

### 1.8.1. Exemple1 :

Si on veut supprimer un employé dans la table « employés » on suit la syntaxe suivante :

```
DELETE FROM employés
WHERE id = 102;
```

### 1.8.2. Exemple 2 :

Si on veut faire une suppression conditionnelle (salaire inférieur à 2000) on suit la syntaxe suivante :

```
DELETE FROM employés
WHERE salaire < 2000;
```

## 1.9. SQL UNION

L'opérateur UNION en SQL permet de combiner les résultats de plusieurs requêtes SELECT en un seul ensemble de résultats, tout en éliminant automatiquement les enregistrements en double. UNION effectue un filtrage distinct, garantissant que chaque ligne du résultat final est unique. Pour fonctionner, les requêtes combinées doivent avoir le même nombre de colonnes, avec des types de données compatibles entre les colonnes correspondantes. La syntaxe de base utilisée pour l'opérateur UNION est :

```
SELECT colonne1, colonne2 FROM table1
UNION
SELECT colonne1, colonne2 FROM table2;
```

## 1.10. SQL UNION ALL

L'opérateur SQL `UNION ALL` permet de rassembler les résultats de plusieurs requêtes `SELECT` en un seul ensemble. Contrairement à l'opérateur `UNION`, il ne supprime pas les doublons et garde toutes les lignes, même si elles sont identiques. Cela le rend plus rapide à exécuter et nécessite moins de puissance de traitement.

Sa syntaxe de base comme suit :

```
SELECT colonne1, colonne2 FROM table1
UNION ALL
SELECT colonne1, colonne2 FROM table2;
```

Le résultat de cette requête affiche un nombre total d'enregistrements égal à la somme de ceux présents dans les deux tables. Cela signifie que certains employés apparaissent dans les deux tables d'origine, et donc, ils sont présents deux fois dans le résultat final. [4]

## 1.11. SQL INSERT INTO

La commande INSERT INTO permet d'ajouter de nouveaux enregistrements dans une table SQL. C'est l'une des opérations fondamentales du langage SQL pour peupler une base de données. Plusieurs syntaxes qu'on utilise pour l'effectuer parmi ces dernières [4] :

— **Insertion simple avec valeurs explicites :**

```
INSERT INTO nom_table (colonne1, colonne2, ...)
VALUES (valeur1, valeur2, ...);
```

— **Insertion multiple en une seule commande :**

```
INSERT INTO nom_table (colonne1, colonne2, ...)
VALUES
    (valeur1, valeur2, ...),
    (valeur1, valeur2, ...),
    ...;
```

— **Insertion à partir d'une requête SELECT :**

```
INSERT INTO nom_table (colonne1, colonne2, ...)
SELECT colonne1, colonne2, ...
FROM autre_table
WHERE condition;
```

## 1.12. SQL INTERSECT

L'opérateur SQL INTERSECT sert à afficher uniquement les résultats qui sont présents dans deux requêtes SELECT ou plus. Autrement dit, il ne garde que les enregistrements communs, comme dans une intersection en mathématiques entre deux ensembles de données. La commande de base est comme suit :

```
SELECT colonnes FROM table1
INTERSECT
SELECT colonnes FROM table2;
```

## 2. Le langage SQL dans MYSQL

MySQL est un système de gestion de bases de données relationnelles (SGBDR) open source, très utilisé dans le développement d'applications web, mobiles et logicielles. Il repose sur un modèle relationnel, dans lequel les données sont organisées sous forme de tables composées de lignes et de colonnes. MySQL se distingue par sa fiabilité, sa rapidité d'exécution, sa compatibilité avec de nombreux langages de programmation (comme PHP, Java ou Python) et sa facilité d'intégration dans divers environnements de développement.
Au cœur du fonctionnement de MySQL se trouve le langage SQL (Structured Query Language), un langage standardisé servant à interagir avec les bases de données. SQL permet aux utilisateurs de créer des bases de données, de définir des structures de tables (CREATE TABLE), d'ajouter des données (INSERT INTO), de les modifier (UPDATE), de les supprimer (DELETE), ainsi que de les interroger (SELECT). Dans MySQL, l'utilisation du SQL est optimisée pour assurer une exécution rapide des requêtes, même sur de grandes quantités de données. De plus, le langage permet d'appliquer des contraintes pour garantir l'intégrité des données (comme PRIMARY KEY, FOREIGN KEY, ou NOT NULL), et offre des fonctionnalités avancées comme les jointures (JOIN), les sous-requêtes, les vues (VIEW) et la gestion des transactions (START TRANSACTION, COMMIT, ROLLBACK).
Grâce à cette richesse fonctionnelle, le langage SQL dans MySQL permet de concevoir des bases de données efficaces, bien structurées et sécurisées, adaptées à une grande variété de projets. Il constitue ainsi un outil incontournable pour tout développeur ou administrateur de bases de données souhaitant gérer l'information de manière rigoureuse et professionnelle. [6] [15]

# Chapitre 3

# Interroger une base de données

Aujourd'hui, les données sont partout. Que ce soit dans les hôpitaux, les entreprises ou même les administrations, savoir chercher et trouver les bonnes informations dans une base de données est devenu indispensable. Imaginez un médecin qui doit identifier rapidement les patients à risque, ou un logisticien qui doit optimiser ses livraisons. Une simple requête bien faite peut leur faire gagner un temps précieux et leur éviter des erreurs. Dans ce chapitre, nous allons voir comment interroger efficacement une base de données. Pas besoin d'être un expert en informatique : avec les bonnes méthodes, tout le monde peut apprendre à trouver exactement ce qu'il cherche, rapidement et sans se tromper. [21]

## 1. Connexion a la base de données

Avant de pouvoir interagir avec une base de données, il faut d'abord établir une connexion, comme on aurait besoin d'une clé pour ouvrir une porte.
Cette étape technique mais cruciale nécessite quelques informations de base : l'adresse où se trouve la base (généralement "localhost" pour une installation locale), un nom d'utilisateur et un mot de passe valides, ainsi que le nom précis de la base à laquelle on souhaite accéder. La connexion peut se faire soit via une interface graphique intuitive (comme phpMyAdmin) où l'on remplit simplement un formulaire, soit programmatiquement en utilisant des fonctions spécifiques dans des langages comme PHP ou Python. Il arrive fréquemment que des problèmes surviennent à cette étape « mot de passe incorrect, permissions insuffisantes ou service non démarré » qu'il faut alors résoudre avant de pouvoir continuer. Cette connexion initiale est le passage obligé avant toute manipulation des données. Elle doit être sécurisée pour protéger les informations sensibles. Une fois réussie, on obtient enfin accès à toutes les fonctionnalités de la base de données. [17]

## 2. La projection

La commande SELECT effectue une projection, qui est une étape essentielle dans une requête SQL qui consiste à sélectionner uniquement certaines colonnes d'une table plutôt que de récupérer toutes les colonnes. Elle permet de limiter les données retournées pour améliorer les performances et la lisibilité des résultats.

### 2.1. Syntaxe

```
SELECT colonne1, colonne2, ...
FROM nom_table;
```

— Exemple

```
SELECT nom, prenom, departement
FROM employes;
```

## 2.2. Fonctions d'agrégations :

Les fonctions d'agrégation effectuent des calculs sur un ensemble de lignes et retournent une valeur unique. Les principales fonctions sont [12] :

| Fonctions | Description | Exemple | Résultat |
|---|---|---|---|
| COUNT() | Compte le nombre de lignes | SELECT COUNT(*) FROM employes ; | Nombre total d'employés |
| SUM() | Fait la somme des valeurs | SELECT SUM(salaire) FROM employes ; | Somme des salaires |
| AVG() | Calcule la moyenne | SELECT AVG(salaire) FROM employes ; | Salaire moyen |
| MIN()/MAX() | Retourne la valeur minimale / maximale | SELECT MAX/-MIN(salaire) FROM employes ; | Salaire minimum / maximum |

TABLE 3.1 – Exemples de fonctions d'agrégation en SQL

## 2.3. Compléments

L'opérateur * (étoile) permet d'afficher tous les champs d'une table. Il simplifie et accélère l'écriture des requêtes, car sans lui, il serait nécessaire de mentionner individuellement chaque champ.

— **Exemple**

```
SELECT * FROM employes;
```

(Signifie donne l'ensemble des champs de la table employes).

L'opérateur AS permet de donner un nom à un champ d'une requête.

— **Exemple**

```
SELECT nom AS Nom, prenom AS Prénom
FROM employes;
```

# 3. La sélection (Restriction )

La sélection, aussi appelée restriction, permet de filtrer les lignes d'une table en fonction de certaines conditions précises. Par exemple, si l'on veut afficher uniquement les employés dont le salaire est supérieur à 2000 euros, on utilise la commande WHERE pour définir cette condition. Contrairement à la projection, qui consiste à choisir quelles colonnes (ou champs) afficher, la sélection s'intéresse aux lignes à conserver dans le résultat. [16]

## 3.1. Syntaxe

```
SELECT colonnes FROM table WHERE condition;
```

— Exemple :

```
SELECT nom, prenom, salaire
FROM employes
WHERE departement = 'informatique'
```

Ceci affiche les noms et prénoms et le salaire des employés dans le département informatique.

— Exemple 2

```
SELECT nom, salaire
FROM employes
WHERE salaire > 2000;
```

Ceci affiche les noms et le salaire des employés qui gagne plus de 2000 euros.

## 3.2. Les opérateurs

Les opérateurs de restriction (WHERE) permettent de filtrer les lignes selon des conditions spécifiques :

— `=` (égal) et `!=` / `<>` (différent) comparent des valeurs.

— `>`, `<`, `>=`, `<=` comparent des nombres ou des dates.

— `BETWEEN` vérifie si une valeur est dans un intervalle
*Exemple :* `salaire BETWEEN 3000 AND 5000`.

— `LIKE` recherche des motifs textuels
*Exemple :* `nom LIKE 'Mar%'` pour les noms commençant par `"Mar"`.

— `IN` teste si une valeur est dans une liste
*Exemple :* `departement IN ('IT', 'RH')`.

— `IS NULL` identifie les valeurs manquantes
*Exemple :* `date_depart IS NULL`.

— `AND` / `OR` permettent de combiner plusieurs conditions.

Ces opérateurs s'utilisent après WHERE pour affiner les résultats

# 4. La jointure

La jointure permet de récupérer des données provenant de plusieurs tables en les reliant entre elles. Cette liaison se fait généralement grâce à une clé étrangère, qui établit un lien entre les enregistrements des différentes tables. Lorsque la commande FROM mentionne plusieurs tables, il faut indiquer au moins un critère de jointure pour préciser comment ces tables sont liées. Ce critère est spécifié dans la clause WHERE. S'il y a plusieurs conditions de jointure, on utilise l'opérateur AND pour les combiner. [3]

### 4.1. Exemple

```
SELECT employes.nom, departements.nom
FROM employes
WHERE employes.id_dept = departements.id;    ;
```

Cette requête affiche le nom de chaque employé ainsi que le nom de son département, en reliant les deux tables grâce à la clé id-dept dans la table employes qui correspond à l'identifiant du département dans la table departements.

## 5. Le TRI (ORDER BY)

La clause ORDER BY permet d'ordonner les résultats d'une requête SQL selon une ou plusieurs colonnes, par ordre croissant (ASC) ou décroissant (DESC).

### 5.1. Syntaxe

```
SELECT colonne1, colonne2
FROM table
WHERE conditions
ORDER BY colonne1 ASC, colonne2 DESC;
```

### 5.2. Exemple

```
SELECT nom, salaire
FROM employes
ORDER BY salaire DESC;
```

Cette requête affiche le nom et le salaire des employés, puis trie les résultats par salaire, du plus élevé au plus bas grâce à ORDER BY salaire DESC (DESC signifie décroissant).

## 6. Le regroupement

### 6.1. La commande GROUP BY

La clause GROUP BY en SQL sert à regrouper les lignes qui ont des valeurs communes dans une ou plusieurs colonnes. Elle est souvent utilisée avec des fonctions comme COUNT, SUM ou AVG pour faire des calculs (comme des totaux ou des moyennes) sur chaque groupe. C'est utile quand on veut faire des résumés par catégorie, comme le total des ventes par produit ou le nombre d'employés par service. [16]

### 6.2. Syntaxe

```
SELECT colonne1, fonction_agrégat(colonne2)
FROM table
WHERE condition
GROUP BY colonne1;
```

### 6.3. Exemple

```
SELECT departement, COUNT(*)
FROM employes
GROUP BY departement;
```

Cette requête compte le nombre d'employés dans chaque département. Elle regroupe les lignes par valeur de la colonne département, puis utilise COUNT(*) pour compter les employés dans chaque groupe.

## 7. La restriction sur le regroupement

La clause HAVING permet de filtrer les groupes après qu'ils ont été créés avec GROUP BY. Contrairement à WHERE qui filtre les lignes avant le regroupement, HAVING sert à garder seulement les groupes qui respectent une certaine condition.

### 7.1. Syntaxe

```
SELECT colonne1, fonction_agrégat(colonne2)
FROM table
WHERE conditions
GROUP BY colonne1
HAVING condition_sur_agrégat;
```

### 7.2. Exemple

```
SELECT departement, COUNT(*) AS nb_employes
FROM employes
GROUP BY departement
HAVING COUNT(*) > 5;
```

Cette requête affiche uniquement les départements comptant plus de 5 employés. En résumé, SQL est un langage qui sert notamment à consulter et modifier une base de données à l'aide de différentes commandes.

- La commande SELECT...FROM permet de choisir les colonnes à afficher, c'est ce qu'on appelle une projection.
- La commande WHERE sert à filtrer les lignes selon une condition, c'est ce qu'on appelle une sélection (ou restriction).
- Quand on travaille avec plusieurs tables, il faut faire une jointure, généralement exprimée aussi avec WHERE pour relier les données.
- Pour trier les résultats, on utilise ORDER BY.
- Pour regrouper des données similaires, on utilise GROUP BY, souvent combiné avec des fonctions comme COUNT, SUM ou AVG.
- Enfin, si on veut filtrer ces groupes, on ajoute HAVING après le regroupement.

# Chapitre 4

# REQUÊTES

Les requêtes SQL permettent d'extraire et manipuler des données dans une base. La structure de base comprend SELECT pour choisir les colonnes, obligatoirement suivi de FROM pour spécifier la table source. On peut ensuite ajouter WHERE pour filtrer, utiliser des opérateurs (arithmétiques $+ - */$ ou de comparaison $=, <>$) et enfin organiser les résultats avec ORDER BY. Ce chapitre expliquera comment combiner ces éléments pour créer des requêtes efficaces.[20]

## 1. Opérateurs Arithmétiques

Les opérateurs arithmétiques en SQL permettent d'effectuer des calculs mathématiques directement dans les requêtes. Ils s'appliquent aux colonnes numériques (INT, FLOAT, DECIMAL, etc.) et sont essentiels pour manipuler des données chiffrées sans traitement externe. Voici les différents opérateurs [14] :

| Opérateurs | Description |
|---|---|
| + | Addition |
| − | Soustraction |
| * | Multiplication |
| / | Division |
| % (MOD) | Modulo (reste de division) |

TABLE 4.1 – Les opérateurs arithmétiques

### 1.1. Exemple 1

```
SELECT salaire + 200 AS salaire_avec_bonus FROM employes;
```

Cette syntaxe veut dire qu'on va ajouter 200 DA au salaire de chaque employé. Le résultat affiche une nouvelle valeur appelée salaire-avec-bonus.

### 1.2. Exemple 2

```
SELECT salaire - 150 AS salaire_apres_penalite FROM employes;
```

Cette syntaxe veut dire soustrait 150 DA du salaire de chaque employé pour simuler une pénalité. Le résultat montre le salaire réduit.

### 1.3. Exemple 3

```
SELECT salaire / 30 AS salaire_par_jour FROM employes;
```

Cette syntaxe veut dire divise le salaire mensuel par 30 pour estimer le salaire par jour.

## 2. Opérateurs de comparaison

Les opérateurs de comparaison sont essentiels pour filtrer et analyser des données en SQL. Ils permettent de définir des conditions dans les clauses WHERE, HAVING ou JOIN pour sélectionner uniquement les enregistrements qui répondent à des critères spécifiques. [19]

| **Opérateurs** | **Descriptions** |
|---|---|
| = | Égale à (égalité de deux expressions). |
| <> ou ! = | Différent de (détermine si l'expression de gauche n'est pas égale à celle de droite). |
| < | Inférieure (compare deux expressions pour déterminer si la valeur de l'expression de gauche est inférieure à celle de l'expression de droite). |
| > | Supérieure (compare deux expressions pour déterminer si la valeur de l'expression de gauche est supérieure à celle de l'expression de droite). |
| ≤ | Inférieure ou égale (compare deux expressions pour déterminer si la valeur de l'expression de gauche est inférieure ou égale à celle de l'expression de droite). |
| ≥ | Supérieure ou égale (compare deux expressions pour déterminer si la valeur de l'expression de gauche est supérieure ou égale à celle de l'expression de droite). |
| BETWEEN | Entre deux valeurs (inclusives). |
| IN | Dans une liste de valeurs. |
| IS NULL / IS NOT NULL | Vérifie si une valeur est NULL ou pas. |
| LIKE | Correspond à un modèle (texte). |

TABLE 4.2 – Opérateurs de comparaison en SQL

### 2.1. Exemple 1

```
SELECT * FROM employes WHERE salaire >= 3000;
```

Va afficher les employés dont le salaire est supérieur ou égal à 3000 DA.

### 2.2. Exemple 2

```
SELECT * FROM employes WHERE date_embauche < '2022-01-01';
```

Va afficher les employés embauchés avant le **1er janvier 2022**.

### 2.3. Exemple 3

```
SELECT * FROM employes WHERE poste <> 'Secrétaire';
```

Va afficher tous les employés sauf ceux qui occupent le poste de Secrétaire.

### 2.4. Exemple 4

```
SELECT * FROM employes WHERE diplome = 'Master';
```

Va afficher les employés qui ont un Master comme diplôme.

## 3. Opérateurs logiques et cas d'expression

Les opérateurs logiques permettent de combiner plusieurs conditions dans vos requêtes SQL pour créer des filtres complexes. Ils sont essentiels pour affiner vos recherches et analyses de données. [13]

| Opérateurs | Définitions |
|---|---|
| AND (logique) | Retourne vrai si toutes les conditions sont vraies. |
| OR (logique) | Retourne vrai si au moins une condition est vraie. |
| NOT (logique) | inverse le résultat de la condition (vrai → faux, faux → vrai). |
| () | Définit l'ordre d'évaluation des conditions (priorité forcée). |
| THEN | Résultat d'une clause When lorsqu'elle prend la valeur true. |
| CASE | Évalue un ensemble d'expressions booléennes pour déterminer le résultat. |

TABLE 4.3 – Opérateurs logiques

# 4. Opérateurs de requête :

Les opérateurs de requête servent à construire des expressions permettant d'extraire des données liées aux entités. Le tableau suivant répertorie les opérateurs de requête : [9]

| Opérateurs | Utilisations |
|---|---|
| FROM | Spécifie la collection qui est utilisée dans les instructions Select |
| GROUP BY | Spécifie les groupes dans lesquels les objets retournés par une expression de requête (Select) doivent être placés. |
| GroupPartition | Retourne une collection de valeurs d'argument, projetées en dehors de la partition de groupe à laquelle l'agrégat est associé. |
| HAVING | Indique un critère de recherche pour un groupe ou une fonction d'agrégation. |
| SEPAR | Utilisé avec la clause order by pour effectuer la pagination physique |
| ORDER BY | Spécifie l'ordre de tri utilisé sur les objets retournés dans une instruction Select. |
| SELECT | Spécifie les éléments dans la projection qui sont retournés par une requête |
| SAUT | Utilisé avec la clause order by pour effectuer la pagination physique. |
| TOP | Spécifie que seul le premier ensemble de lignes sera retourné à partir du résultat de la requête. |
| WHERE | Filtre de manière conditionnelle les données qui sont retournées par une requête. |

TABLE 4.4 – Opérateurs de requête

# 5. Autre opérateurs

Voici des autres opérateurs :

| Opérateurs | Utilisation |
|---|---|
| IN / NOT IN | Appartenance où non à un ensemble N. |
| ALL | Comparaison à un ensemble. |
| LIKE / NOT LIKE | Permettent de rechercher des chaînes de caractères selon un motif (trouver/exclure) |
| NULL | Pas de valeur |
| Expression logique | combinaisons entre les opérateurs logiques. |

TABLE 4.5 – Autre Opérateurs

# Chapitre 5

# la partie pratique

## 1. Présentation de l'ENIEM

### 1.1. Situation géographique

L'ENIEM (Entreprise nationale des industriels et de l'électroménager), est une entreprise publique et économique de droit Algérien (EPE). Son siège social se situe à la wilaya de **TIZI-OUZOU**, Les unités de production : Froid, cuisson et climatisation sont implantées à la zone industrielle **AISSA AT IDIR** à **Oued Aissi**, distance de 10 Km de la ville de **TIZI OUZOU**, elle s'étale sur une superficie de 55 hectares et elle relève administrativement de la commune **L'Arebaa-Nait-Iraten**. La filiale sanitaire est installée à **Miliana**, wilaya **Ain Defla**, et la filiale lampe à **Mohammadia**, wilaya de **Mascara**.

### 1.2. Historique

L'entreprise National des industriels et de L'électroménager (ENIEM), constituée le **2 janvier 1983**, bien qu'elle ait été fondée en **1974** en tant que branche de l'entreprise SONELEC. L'ENIEM a été chargé de la production et commercialisation des produits électroménager et disposait à sa création de :
* Un complexe d'appareil ménager (CAM) à **Tizi Ouzou**, entré en production et la commercialisation des produits en **juin 1977**. * Une entreprise de fabrication de lampes à **Mohammadia**, entré en production en **février 1979**. * Une entreprise de fabrication des produits sanitaires à **Miliana**, entré en production en **1979**.

### 1.3. Mission de l'ENIEM

L'entreprise ENIEM est leader de l'électroménager en Algérie, elle possède des capacités de production et une expérience de 30 ans dans la fabrication et développement et la commercialisation des appareils électroménagers notamment :

— Les appareils ménagers domestiques.

— Les appareils de collectivité.

— Les appareils d'éclairages.

— Les produits sanitaires.

## 1.4. Objectif de l'entreprise

L'ENIEM vise les objectifs suivants :

— L'amélioration de la qualité des produits.
— La maîtrise des coûts de production.
— L'augmentation des capacités, d'étude et développement.
— L'amélioration de la maintenance de l'outil de production des installations.
— La valorisation des ressources humaines.
— L'augmentation du volume de production.
— Le renforcement de la sécurité du patrimoine et des installations.
— La réduction du coût de non qualité.

## 1.5. Mode d'organisation

L'organisation de l'ENIEM est structurée comme suit

### 1.5.1. organigramme de l'ENIEM

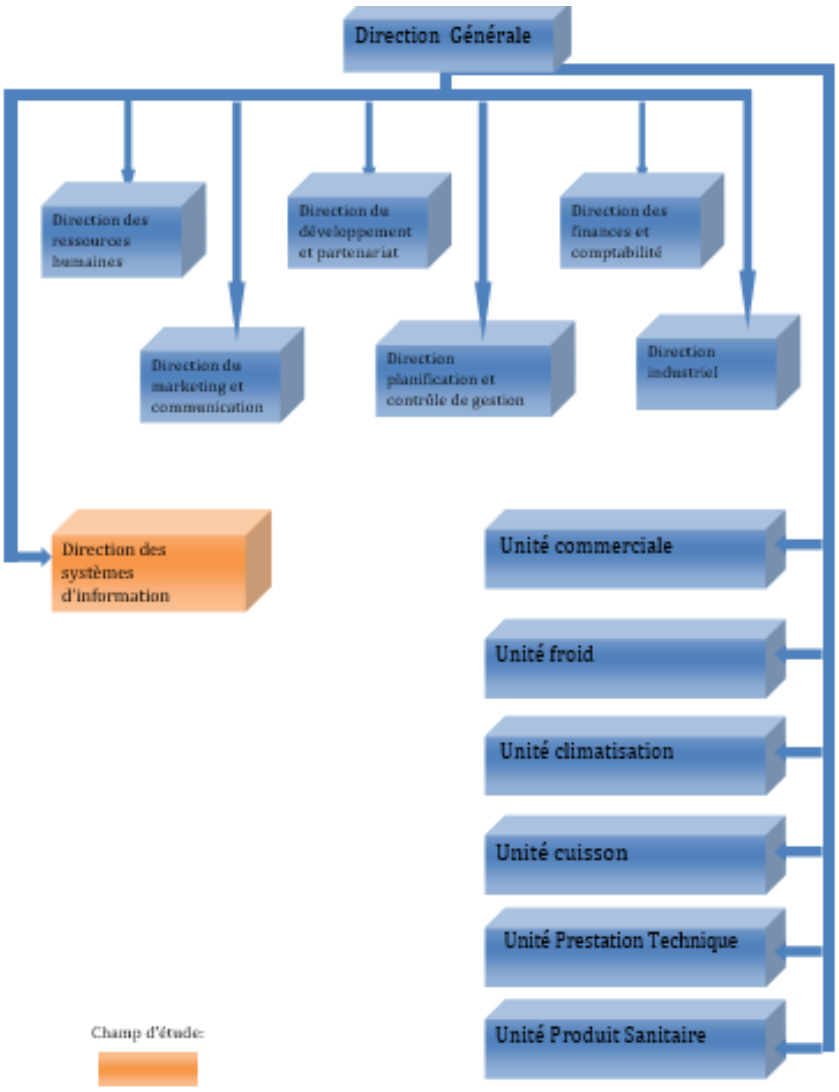


FIGURE 5.1 – organigramme de l'ENIEM

Afin de mieux comprendre l'organisation de l'ENIEM, il est essentiel d'examiner en détail chaque structure et son rôle en sein de l'entreprise. Voici une présentation de différente directions et les différentes unités et leur attribution

### 1.5.2. Les directions

ENIEM est dotée d'une direction générale et de sept directions Centrales

**(a) Direction Générale :**la direction générale est l'unique entité responsable de la stratégie et du développement de l'entreprise. Elle exerce son autorité hiérarchique et fonctionnelle sur l'ensemble des directions et des unités.

**(b) Direction des ressources humaines :** en cohérence avec la politique qualité de l'entreprise, la fonction ressources humaines accroît la mobilisation et la valorisation du personnelle dans ses actions au service du client. Elle pilote le recrutement, l'accueil, l'information et gère le plan de carrière du personnel, elle supervise la gestion administrative et légale pour le personnel.

**(c) Direction du marketing et de la communication :** La direction du marketing et de la communication assure des politiques commerciales et des communications et les met en œuvre par la conception et l'élaboration des méthodes et outils de gestion nécessaires.

**(d) Direction de développement et partenariat :** cette direction assure l'étude et le développement du produit fini ainsi que des actions de partenariat et de sous6traitance.

**(e) Direction planification et contrôle de gestion :** cette direction est responsable du contrôle de gestion, de l'audit finance ainsi que des activités liées au budget de l'entreprise notamment la préparation, l'établissement et le suivi. Elle exploite les résultats de l'audit finance, les interprète, et fait les recommandations nécessaires.

**(f) Direction des finances et comptabilités :** les fonctions qu'assure cette direction sont principalement :

- Analyser les équilibres financiers de l'entreprise.
- Gérer la trésorerie (recettes et dépenses).
- Analyser les coûts et prix de revient et mettre à la disposition des responsables opérationnelles l'information financière nécessaire.

**(g) Direction industrielle** la direction industrielle est chargée de développer et de mettre en place les moyens et l'organisation industrielle nécessaire à la production en agissant sur les approvisionnements, les moyens et les techniques de production.

**(h) Direction des systèmes d'information (DSI) :** La direction des systèmes d'information (DSI) joue un rôle stratégique au sein de l'entreprise en assurant la gestion, l'optimisation et la sécurisation des systèmes d'information. Elle veille à aligner les technologies de l'information avec les objectifs de l'organisation afin d'améliorer l'efficacité opérationnelle et la compétitivité. Ses missions incluent la conception et la maintenance des infrastructures informatiques, le développement et l'administration des bases de données, ainsi que la gestion des applications métiers. De plus, la DSI garantit la cybersécurité et veille à la conformité aux réglementations en vigueur, tout en accompagnant la transformation digitale de l'entreprise par l'intégration de nouvelles technologies. Ainsi, elle contribue activement à l'innovation et à l'amélioration continue des processus internes.

### 1.5.3. Les unités

En plus des directions, l'entreprise est organisée par centres d'activités stratégiques qui se composent de 05 unités de production, et d'une unité commerciale.

**(a) Unité commerciale :**cette unité est chargée essentiellement d la commercialisation des produits de l'entreprise, de la promotion des exportations et de la gestion du service après-vente.

**(b) Unité froid :** l'unité froid est chargée de produire et de développer les produits de froid domestiques tels que les réfrigérateurs et les congélateurs. Elle dispose d'un laboratoire central composé de laboratoires de chimie : un de métallurgie et un autre d'essais produits, et d'un ensemble d'ateliers assurant différents traitements.

**(c) Unité Climatisation :** cette unité est chargée de produire et de développer les produits de climatisation, de chauffage et autre produits similaires tels que chauffe- eaux, chauffe-bain.... Elle possède quatre ateliers de fabrication.

**(d) Unité cuisson**la mission globale de cette unité est de produire et développer les produits de cuisson à gaz, électriques et mixtes. Elle dispose d'un laboratoire d'essais gazier et de quatre ateliers de fabrications.

**(e) Unité prestation technique (UPT) :** l'UPT, comme son nom l'indique, a pour mission de rendre des services à l'ensemble des unités de l'ENIEM. Elle est divisée en huit départements et quatre services.

**(f) Unités Produits Sanitaires :** l'unité des produits sanitaires a été acquise par ENIEM en l'an 2000. Sa mission globale est de produire et développer les produits sanitaires (baignoires, Lavabos et éviers).

## 1.6. Présentation du champ d'étude

Nous avons été orientés vers le service production informatique (SPI) (au sein de la DSI) pour notre stage pratique et cette partie du travail permettra de mieux définir le domaine d'études et mieux apercevoir ses objectifs.

— **Service production informatique** Le service production informatique, intégré à la Direction des Systèmes d'Information (DSI), joue un rôle essentiel dans la gestion et l'exploitation des bases de données de l'entreprise. Il assure la disponibilité, la performance et la sécurité des systèmes et applications en garantissant leur bon fonctionnement au quotidien. Ses missions incluent la surveillance et la maintenance des serveurs, des bases de données, ainsi que la gestion des mises en production et l'optimisation des ressources informatiques. De plus, il met en place des plans de continuité et de reprise d'activité afin de prévenir les incidents majeurs et d'assurer la protection des données. Grâce à l'automatisation des processus et à l'assistance technique aux utilisateurs, le service production Informatique contribue directement à la fiabilité et à l'efficacité du système d'information de l'entreprise.

### 1.6.1. organigramme du champ d'etude

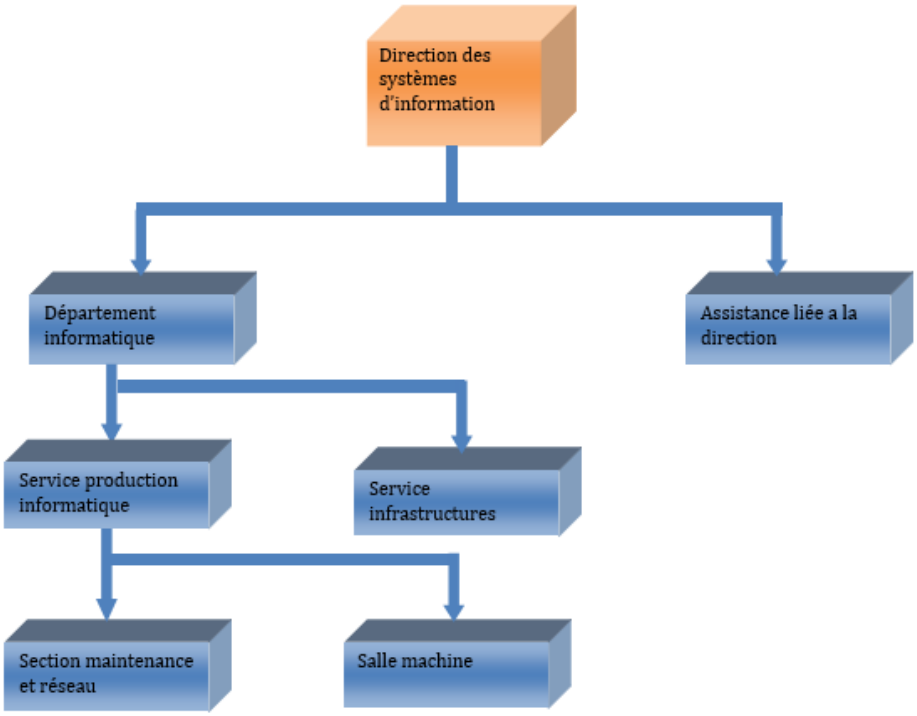


FIGURE 5.2 – Organigramme du champ d'étude

a) **Mission de la DSI :** Superviser les systèmes d'information internes.

I. **Mission de la DSI :**

- Gouvernance des données : assurer la stratégie, la conformité et la sécurité des bases de données.
- Gestion des infrastructures : déployer, configurer et maintenir les serveurs de base de données.
- Sécurité et protection des données : mettre en place des politiques de sauvegarde, de récupération et de protection contre les cyberattaques.
- Optimisation des performances : superviser et améliorer la performance des bases de données.
- Développement et intégration : concevoir des solutions adaptées aux besoins métiers et intégrer les bases de données avec d'autres systèmes d'information.

II. **Activités spécifique liées aux bases de données :**

- Administration des bases de données (installation, maintenance, mise à jour)
- Conception et modélisation des bases de données selon les besoins des utilisateurs.
- Développement et optimisation des requêtes SQL.
- Mise en place des politiques de sauvegarde été de restauration des bases.
- Intégration des bases de données avec les applications de l'entreprise.

III. **Les différents logiciels utilisés dans le SPI :**

- **Cobol :** Langage de programmation avec le quel toutes les applications opérationnelles sont développées sur le HP3000

— **Réflexion X :** est un émulateur d'accès au serveur depuis les différentes postes.
— **Easy :** est une application installée dans le serveur HP3000 pour gérer la comptabilité des différentes unités.
— **SystemMM0909 :** pour la gestion de stock pièces de rechange, et la gestion de production climatiseur.
— **System MMcuiss :** gestion de la production de l'unité cuisson
— **System MMRef :** gestion de production de l'unité Froid.
— **System Achat :** tout ce qui est relatif à la fonction achat et transite.

Dans le Système MM, plusieurs bases de données utilisées, parmi lesquelles deux bases essentielles : IOSDB et EDCDB, qui jouent un rôle central dans la gestion des données du système.

(a) **IOSDB** : c'est une base de données qui est dédiée à l'enregistrement des informations relatives aux articles dans différentes tables, dont les principales sont :
— **IVN-Master :** Contient les données sur les quantités d'articles disponibles en stock
— **Stock-Activité :** Enregistre les mouvements de stock, notamment les entrées en magasin et les consommations
— **IVN-Location :** Permet à suivre la répartition des articles en fonction de leur emplacement

(b) **EDCDB**C'est une base de données dédiée a la gestion des informations des articles a travers des plusieurs tables, notamment :
— **ITEM-Data :** Contient des informations principales des articles, telles que leur code, désignation et leur unité de mesure...
— **AUX-ITEM-Data :** Stockes des données complémentaires relatives aux articles.
— **ITEM-INDEX :** Regroupe des informations essentielles sur les articles déjà enregistrés, ces données étant immuables et ne pouvant être modifiées.

Pour ma problématique j'ai utilisé les tableaux comme ceci :

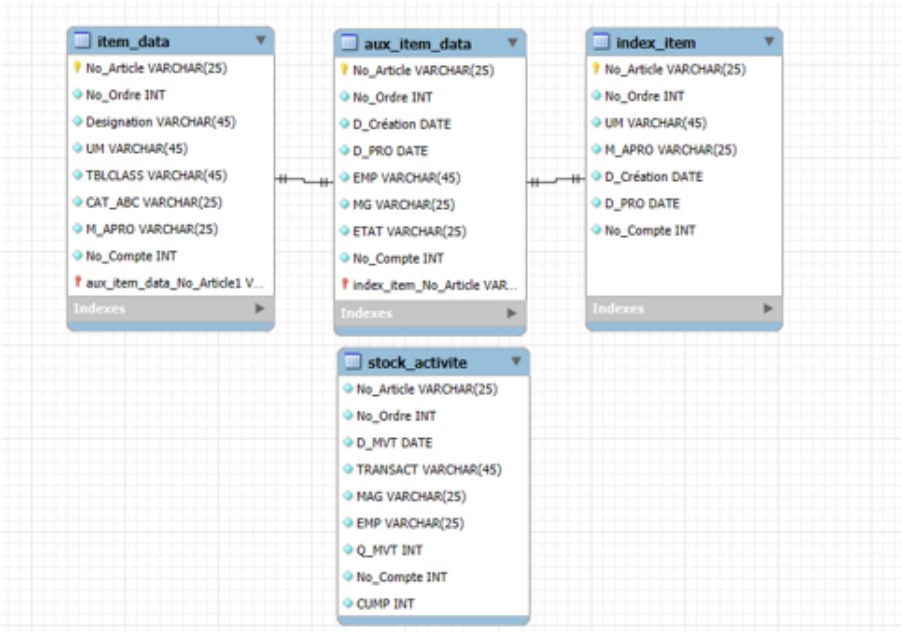


FIGURE 5.3 – Les tables

# 2. Problématique

Dans le cadre de la gestion des flux des données entre le système achats et EASY (comptabilité) au sein de l'ENIEM, plusieurs journaux sont utilisés pour le suivi et l'évaluation des coûts liés aux articles (entré magasin, consommable). Parmi ces journaux, on distingue notamment :

## 2.1. Le journal des encours de réception (813) :

Ce journal regroupe les informations relatives aux articles en cours de réception, ainsi que la conformité de la réception par rapport aux étapes suivantes :

(a) **En cours de réception :**indique que l'article est en phase de réception.

(b) **Contrôle quantitatif :**

   a) Manquant à la réception : en cas de quantité insuffisante constatée lors du contrôle, le contrôleur le signale.

   b) Excédent à la réception : si le contrôle révèle une quantité supérieure à celle commandé, cela est également signalé.

(c) **Contrôle qualitatif**

   a) **Conforme :** confirme que la qualité des articles reçus dépend aux normes requises pour leur utilisation.

   b) **Non conforme :** indique que la qualité des articles reçu ne permet pas leur utilisation.

(d) **Conformité** :rassemble le contrôle qualitatif et le contrôle quantitatif.

(e) **Entrée en magasin :** tous les articles jugés conformes lors du contrôle qualitatif et quantitatif ferons l'objet d'enter magasin vers le stock.

## 2.2. Le journal des mouvements de stock (853) :

Ce journal enregistre toutes les informations relatives aux mouvements des articles en stock. Dans le cas d'unité Froid, il inclut les éléments suivants :

— **Consref :** Suivi de la consommation des articles au sein de l'unité Froid.

— **Entreenmagasin :** Enregistrement des réceptions automatiques (dossier achat) des articles dans le magasin.

— **Cessref :** Suivi des articles cédés à d'autres unités (par exemple vers l'unité cuisson, vers l'unité climatiseur)

— **Receptref :** Enregistrement manuel des articles vers le stock.

Une fois ces journaux sont analysés par le service comptabilité, à travers une interface **SAchat** (une base de données qui se trouve dans le logiciel système achat) vers **EASY**, effectuant un versement des deux journaux cité. Cependant, ce processus rencontre des anomalies bloquantes :

— Absence du numéro de compte comptable : si un compte comptable d'un article n'est pas saisi au niveau de la rubrique « No-Compte » dans la table ITEM-DATA, les transactions associées à cet article ne renseigneront pas le numéro de compte correspondant. Par conséquent, le transfert du journal des mouvements de stock (853) sera entièrement interrompu, entraînant la génération d'une erreur.

— Numéro du compte erroné : si l'article possède un numéro du compte incorrect, celui-ci est systématiquement enregistré au niveau de la transaction de l'article sous la valeur par défaut "399999", ce qui reflète une anomalie dans l'affectation des comptes comptables (versement des journaux vers **EASY**).

### 2.2.1. Proposition de résolution

Pour garantir un transfert fluide et fiable des données entre les systèmes, il est essentiel d'identifier avec précision les articles concernés par ces anomalies, c'est-à- dire sélectionner les articles contenant le numéro du compte erroné, de procéder à la correction des numéros du comptes erroné, et de veiller à ce que chaque article dispose d'une affectation comptable valide avant l'exécution du transfert. C'est-à-dire, qu'une fois l'erreur est détectée, nous avons transmis les numéros des articles concernés au service Finance et Comptabilité afin qu'ils puissent déterminer les comptes comptables correspondants. Une fois les informations reçues, nous avons procédé à la correction nécessaire et relancé le transfert des journaux vers le système EASY. Cela permettra de minimiser les risques d'erreur, d'assurer la traçabilité des données et de renforcer l'intégrité du processus de transfert.

## 2.3. Méthode de résolution de la problématique :

Pour garantir la fiabilité des données dans la création et la gestion d'une base de données, plusieurs méthodes de résolution peuvent être envisagées :

a) **Les jointures :** : permettent de combiner les données de plusieurs tables en comparant les valeurs communes, facilitant ainsi la détection des incohérences ou des données manquantes.
b) **Les contraintes d'intégrité (check, foreigen key) :** assurent la validité des données lors de leur insertion ou mise à jour en imposant des règles spécifiques (par exemple, garantir qu'un numéro de compte existe bien dans une table de référence).
c) **Les déclencheurs (TRIGGERS) :** exécutent automatiquement des actions prédéfinies lors d'événements spécifiques (*INSERT*, *UPDATE*, *DELETE*), permettant ainsi d'automatiser la validation ou la correction des données erronées.
d) **Les procédures stockées :** sont des blocs de code SQL réutilisables qui centralisent les opérations complexes de traitement ou de vérification des données.
e) **Les vues (VIEWS) :**créent des tables virtuelles permettant d'afficher uniquement les données valides ou filtrées, ce qui simplifie l'accès aux informations conformes

Pour ma problématique, j'ai choisi d'utiliser principalement la méthode des jointures, car elle permet de comparer efficacement les données de différentes tables, notamment pour identifier les anomalies liées aux numéros de compte des articles, garantissant ainsi un transfert de données fiable entre les systèmes.

# 3. Déroulement de stage pratique

## 3.1. Création de la base de données

Pour la première étape, nous avons défini et créé la base de données selon la structure suivante :

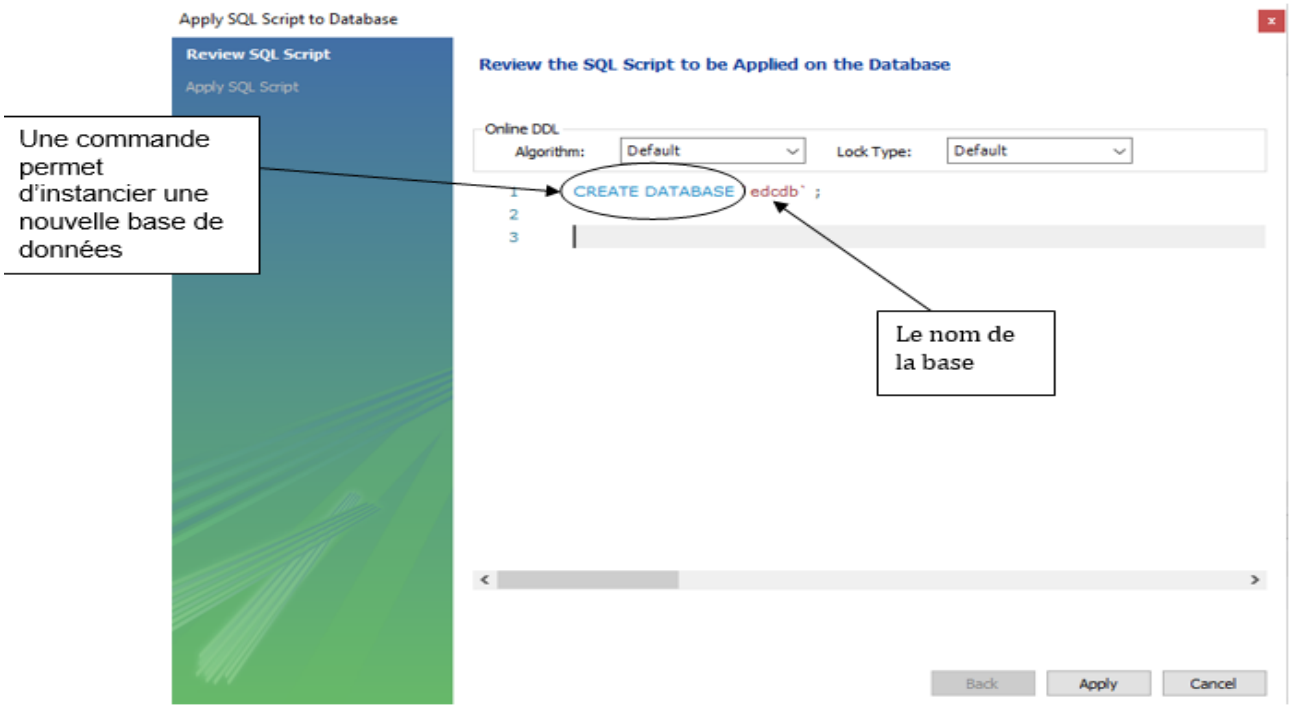


FIGURE 5.4 – Création de la base.

Le résultat est comme suit :

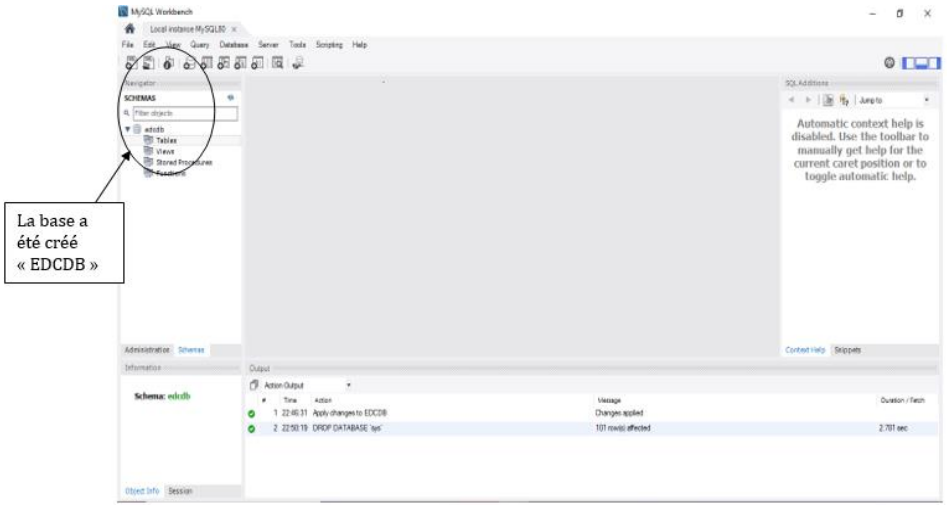


FIGURE 5.5 – la base "EDCDB "

## 3.2. Création des tables

Après avoir défini la base, il est nécessaire d'y structurer les données en concevant des tables adaptées aux besoins du projet. Dans la base de données **EDCDB**, trois tables

essentielles ont été identifiées. Par conséquent, nous procéderons à leur création afin d'assurer la structuration optimale des données. Nous avons créé la première table, intitulée **ITEM-DATA** comprend les colonnes suivantes :

- **No-Article :** Représente le numéro d'identification unique de chaque article.
- **No-Ordre :** Représente le numéro de dossier attribué à chaque article. Ce champ est constitué d'un nombre à six chiffres structuré comme suit : XXYZZZ.
  - XX : correspond à l'année de création du dossier. Pour obtenir l'année réelle, on ajoute 1928 à cette valeur (par exemple, 01 donne 1929).
  - Y : indique l'unité concernée par l'article, selon le code suivant :
    0 : Unité Froid
    3 : Unité Climatisation
    6 : Unité Commerciale
    8 : Unité Cuisson
  - ZZZ : représente le numéro de classement interne, allant de 000 à 999.
- **Desig :**Correspond à la désignation de l'article.
- **UM :** (Unité de mesure) indique l'unité dans laquelle l'article est quantifié.
- **TBCLASSBE :** (Classe technique) définie la classification technique de l'article.
- **CAT-ABC :** (Priorité de l'article) permet de catégoriser les articles en fonction de leur importance
- **M-APRO :** (Mode d'approvisionnement) précise si l'article est fabriqué en interne ou acheté auprès d'un fournisseur.
- **No-Compte :** Désigne le numéro de suivi attribué à chaque article, utilisé pour l'identification comptable et la catégorisation des matières. Ce code permet de reconnaître la nature de l'article selon une nomenclature prédéfinie. Par exemple :
  —
  - 321010 : correspond à un article en plastique.
  - 320017 : correspond à un article en fer.
  - 310016 : désigne également une matière prête à consommer.
  - 321031 : correspond à une matière première.

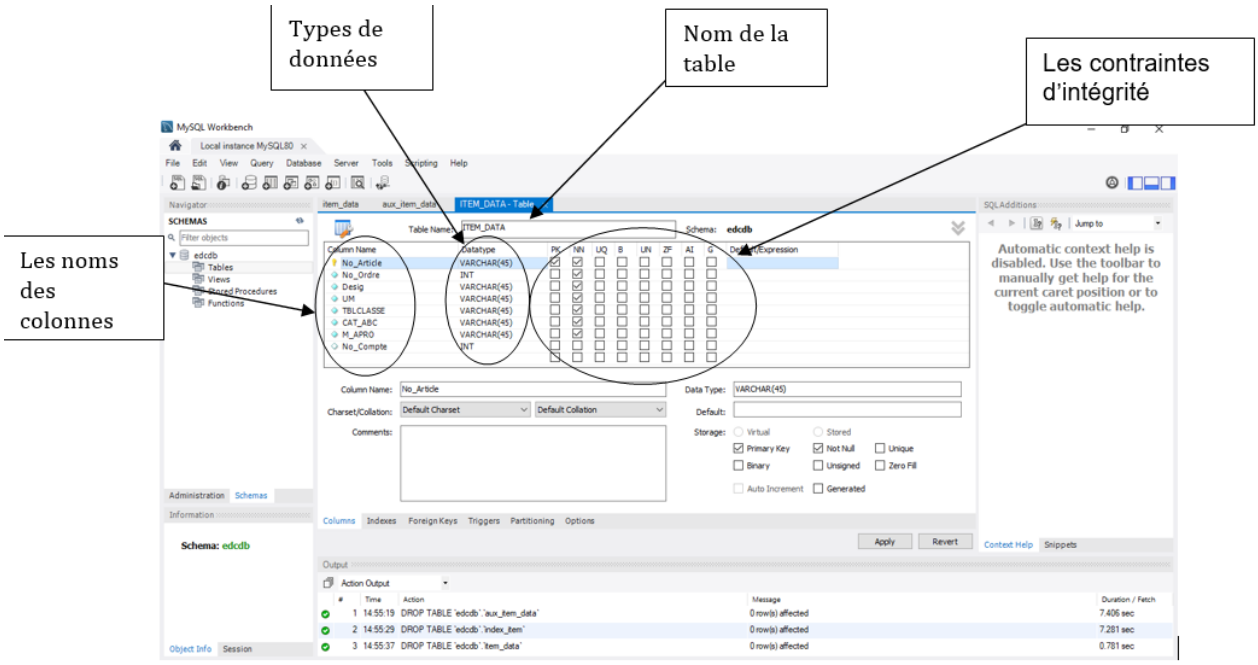


FIGURE 5.6 – Création Table 1 (ITEM-DATA).

En cliquant sur Apply, MySQL affiche :

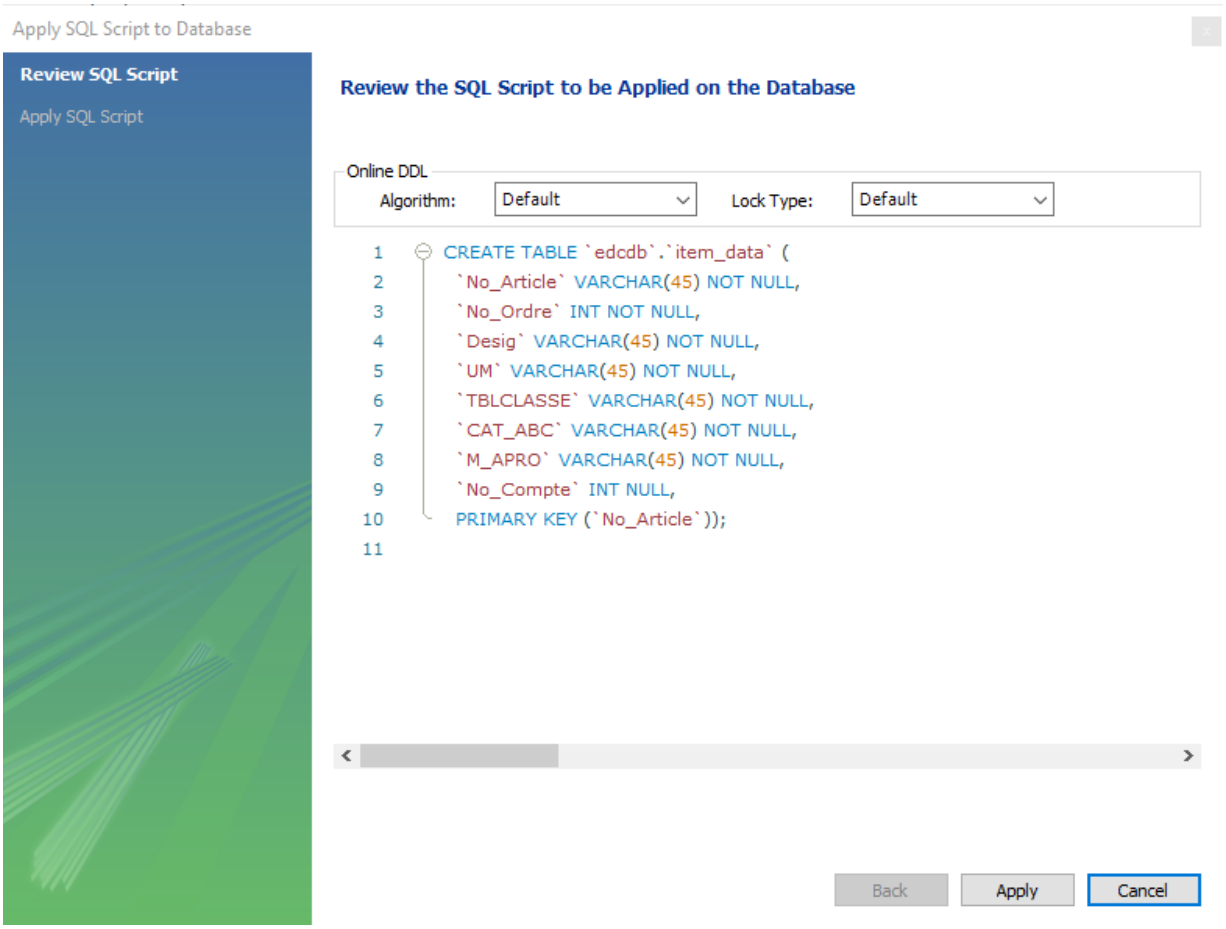


FIGURE 5.7 – Syntaxe création de la table 1

Nous avons exécuté la commande SQL en validant l'opération via l'option Apply nous avons obtenu ceci :

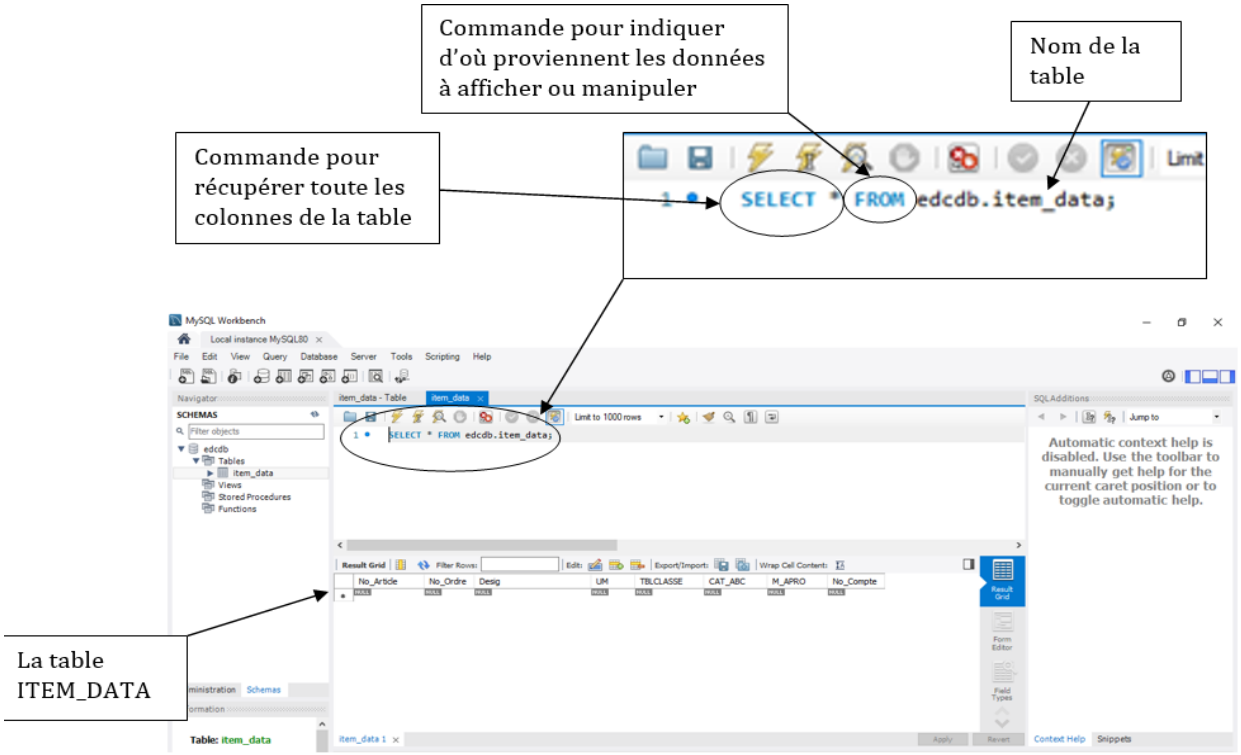


FIGURE 5.8 – Sélection de table 1

Ensuite, pour compléter la table, nous utilisons la syntaxe suivante, illustrée dans la figure ci-dessous : :


FIGURE 5.9 – Données de la table 1

La table s'affiche comme suit :

| No_Article | No_Ordre | Desig | UM | TBLCLASSE | CAT_ABC | M_APRO | No_Compte |
|---|---|---|---|---|---|---|---|
| 140819 | 863203 | JEU DE JOINTS MOTEUR | JEU | R130 | C | F | 321010 |
| 200684 | 860142 | SOUPAPE DE SECURITE | PCE | D120 | B | F | 310034 |
| 258485 | 876020 | BANDE TRANSPORTEUSE | PCE | B9C1 | A | A | 320017 |
| 306845 | 928420 | HUILE TISKA | L | C3000 | A | F | 321010 |
| 402132 | 916230 | KITS DE JOINTS | PCE | BS1D | B | A | 310016 |
| 951357 | 903450 | PORTE FRIGIRATEUR | KG | T4S5 | A | A | 321031 |
| M59753 | 928206 | CONTACT COMPLET | PCE | R460 | A | F | 399999 |
| M64821 | 923130 | PORTE LAMPE | PCE | D821 | C | A | 399999 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

FIGURE 5.10 – La table1 (ITEM-DATA)

Ensuite nous avons créé la deuxième table, intitulé AUX-ITEM-DATA, est composé des colonnes suivantes :

— **No-Article :** identifiant unique de l'article.

— **No-Ordre :** Représente le numéro de dossier attribué à chaque article. Ce champ est constitué d'un nombre à six chiffres structuré comme suit : XXYZZZ.

- XX : correspond à l'année de création du dossier. Pour obtenir l'année réelle, on ajoute 1928 à cette valeur (par exemple, 01 donne 1929).
- Y : indique l'unité concernée par l'article, selon le code suivant :
  0 : Unité Froid
  3 : Unité Climatisation
  6 : Unité Commerciale
  8 : Unité Cuisson
- ZZZ : représente le numéro de classement interne, allant de 000 à 999.

.

— **D-Création :**date de création de l'article.

— **D-PRO :** date de production de l'article.

— **EMP :** emplacement principal attribué à chaque article.

— **MAG :** magasin principal associé à l'article.

— **ETAT :** afficher l'état d'article.

— **No-Compte :** Désigne le numéro de suivi attribué à chaque article, utilisé pour l'identification comptable et la catégorisation des matières. Ce code permet de reconnaître la nature de l'article selon une nomenclature prédéfinie. Par exemple :

—

- 321010 : correspond à un article en plastique.
- 320017 : correspond à un article en fer.
- 310016 : désigne également une matière prête à consommer.
- 321031 : correspond à une matière première.

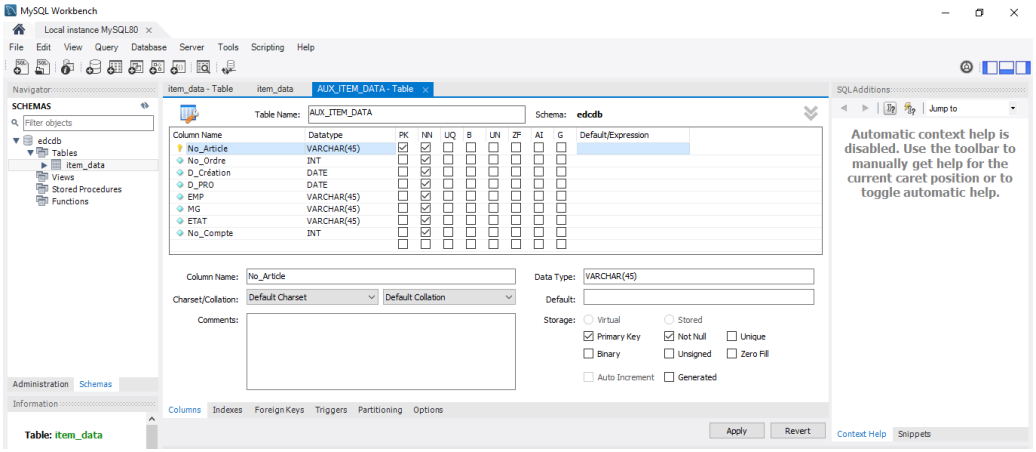


FIGURE 5.11 – Création table 2 (AUX-ITEM-DATA)

En cliquant sur Apply, MySQL affiche :

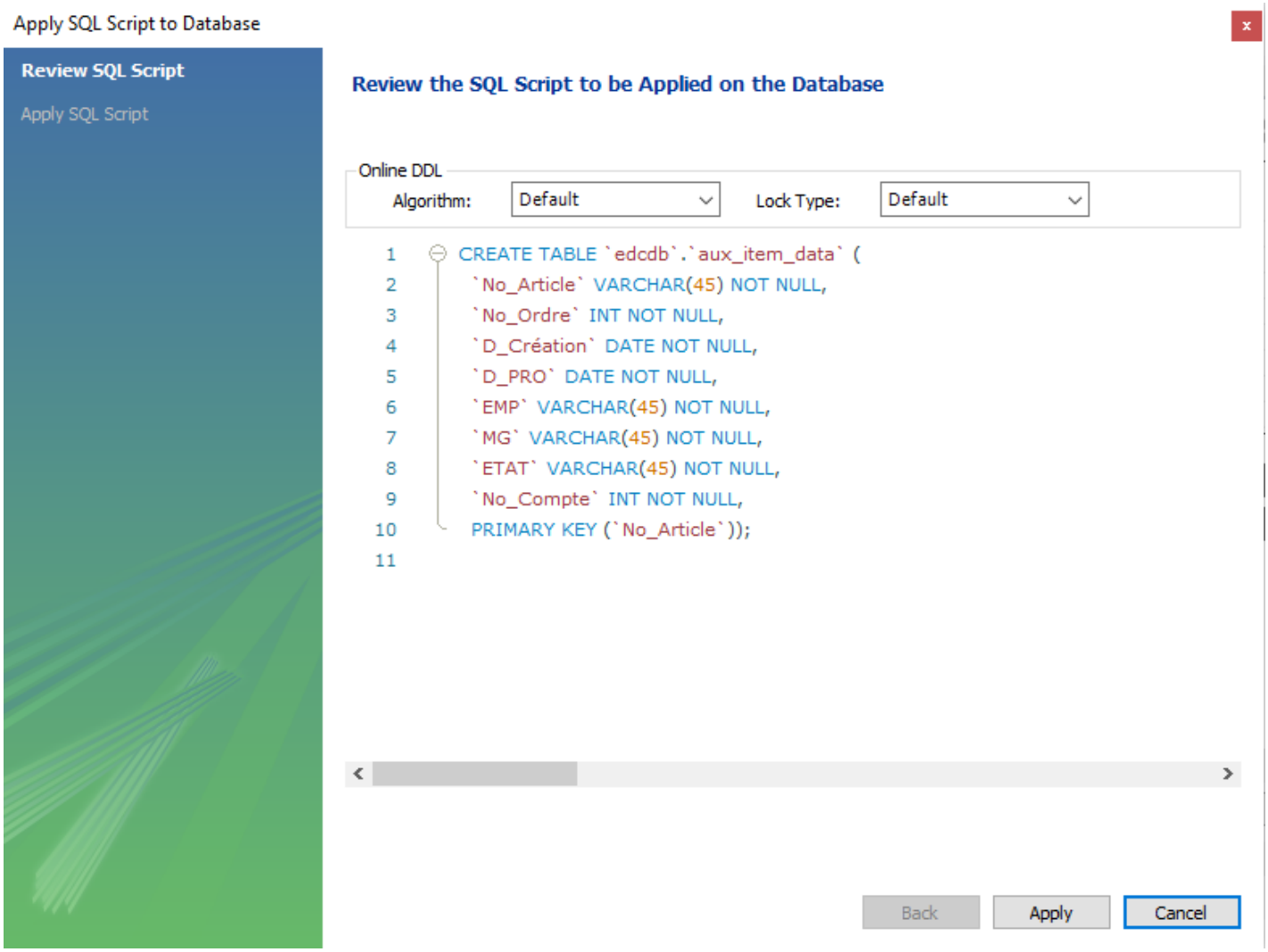


FIGURE 5.12 – Syntaxe création table 2.

Nous avons exécuté la commande SQL en validant l'opération via l'option Apply nous avons obtenu ceci :

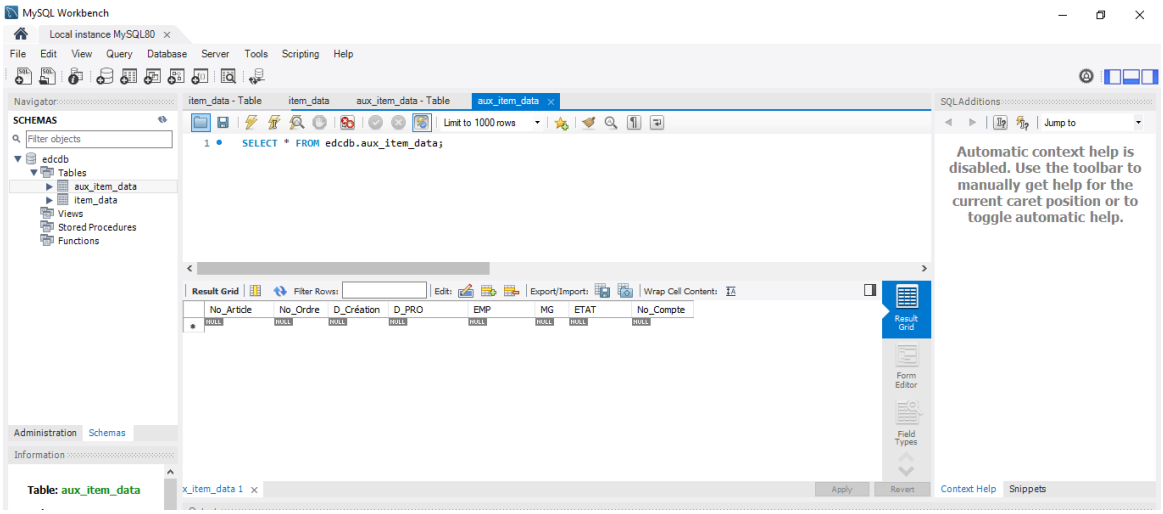


FIGURE 5.13 – Affiche la table 2

Ensuite, pour compléter la table, nous utilisons la syntaxe suivante, illustrée dans la figure ci-dessous :

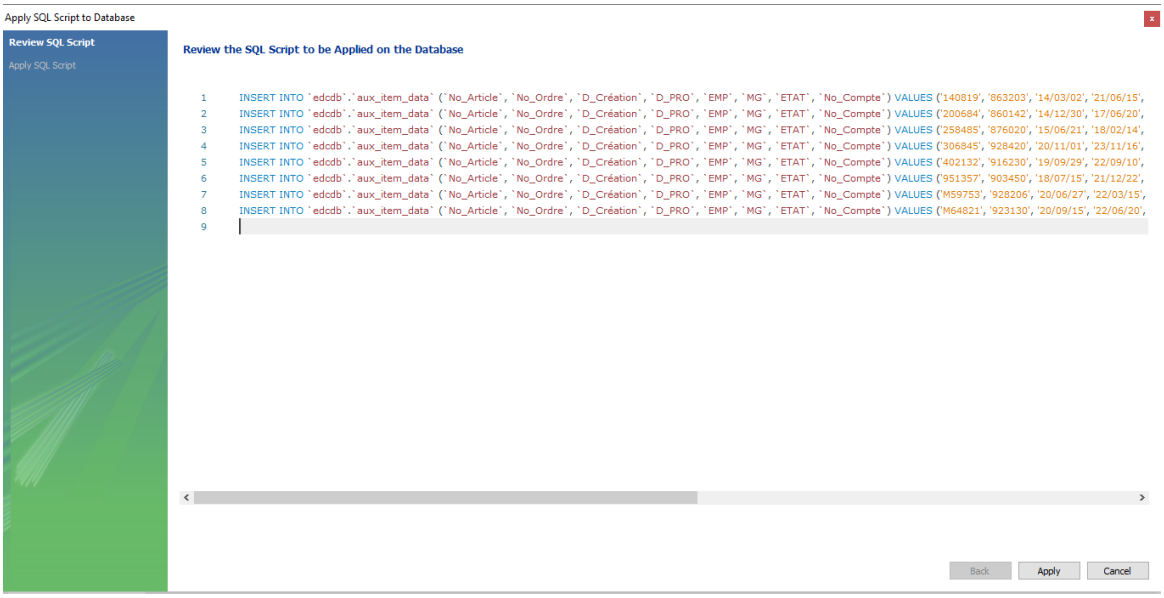


FIGURE 5.14 – Données de la table 2

La table s'affiche comme suit :

| No_Article | No_Ordre | D_Création | D_Pro | EMP | MG | ETAT | No_Compte |
|---|---|---|---|---|---|---|---|
| 140819 | 863203 | 2014-03-02 | 2021-06-15 | A2C4B2 | MI | S | 321010 |
| 200684 | 860142 | 2014-12-30 | 2017-06-20 | B7C9A1 | ME | A | 310016 |
| 258485 | 876020 | 2015-06-21 | 2018-02-14 | D5A3B8 | PR | A | 321031 |
| 306845 | 928420 | 2020-11-01 | 2023-11-16 | F8J4A7 | CR | N | 321010 |
| 402132 | 916230 | 2019-09-29 | 2022-09-10 | E5A4D5 | MJ | S | 310034 |
| 951357 | 903450 | 2018-07-15 | 2021-12-22 | C7A3D8 | CU | A | 320017 |
| M59753 | 928206 | 2020-06-27 | 2022-03-15 | A9B4C8 | RD | S | 399999 |
| M64821 | 923130 | 2020-09-15 | 2022-06-20 | A9B7C2 | FR | N | 399999 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

FIGURE 5.15 – Table2 (AUX-ITEM-DATA)

En suivant les mêmes étapes pour la 3eme table, intitulé INDEX-ITEM on obtient :

| No_Article | No_Ordre | UM | M_APRO | D_Création | D_PRO | No_Compte |
|---|---|---|---|---|---|---|
| 140819 | 863203 | JEU | F | 2014-03-02 | 2021-06-15 | 321010 |
| 200684 | 860142 | PCE | F | 2014-12-30 | 2017-06-20 | 310034 |
| 258485 | 876020 | PCE | A | 2015-06-21 | 2018-02-14 | 320017 |
| 306845 | 928420 | L | F | 2020-11-01 | 2023-11-16 | 321010 |
| 402132 | 916230 | PCE | A | 2019-09-29 | 2022-09-10 | 310016 |
| 951357 | 903450 | KG | A | 2018-07-15 | 2021-12-22 | 321031 |
| M59753 | 928206 | PCE | F | 2020-06-27 | 2022-03-15 | 399999 |
| M64821 | 923130 | PCE | A | 2020-09-15 | 2022-06-20 | 399999 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL |

FIGURE 5.16 – Table 3 (INDEX-ITEM)

Dans la base de données **IOSDB**, parmi les trois tables essentielles précédemment citées, nous avons crée une table, intitulée **Stock-Activite** comprend les colonnes suivantes :

- **No-Article :** Représente le numéro d'identification unique de chaque article.
- **No-Ordre :** Représente le numéro de dossier pour chaque article.
- D-MVT : date de mouvement d'un article (entré magasin, consommer...)
- **TRANSACT :** le type de mouvement pour chaque article (consommé, entre magasin)
- **EMP :** emplacement principal attribué à chaque article.
- **MAG :** magasin principal associé à l'article.
- **Q-MVT :** quantité d'article ayant subit un mouvement.
- **No-Compte :** le numéro de suivis pour chaque article.
- **CUMP :** coût unitaire moyenne pondéré.

D'abord, nous avons crée la base comme suit :

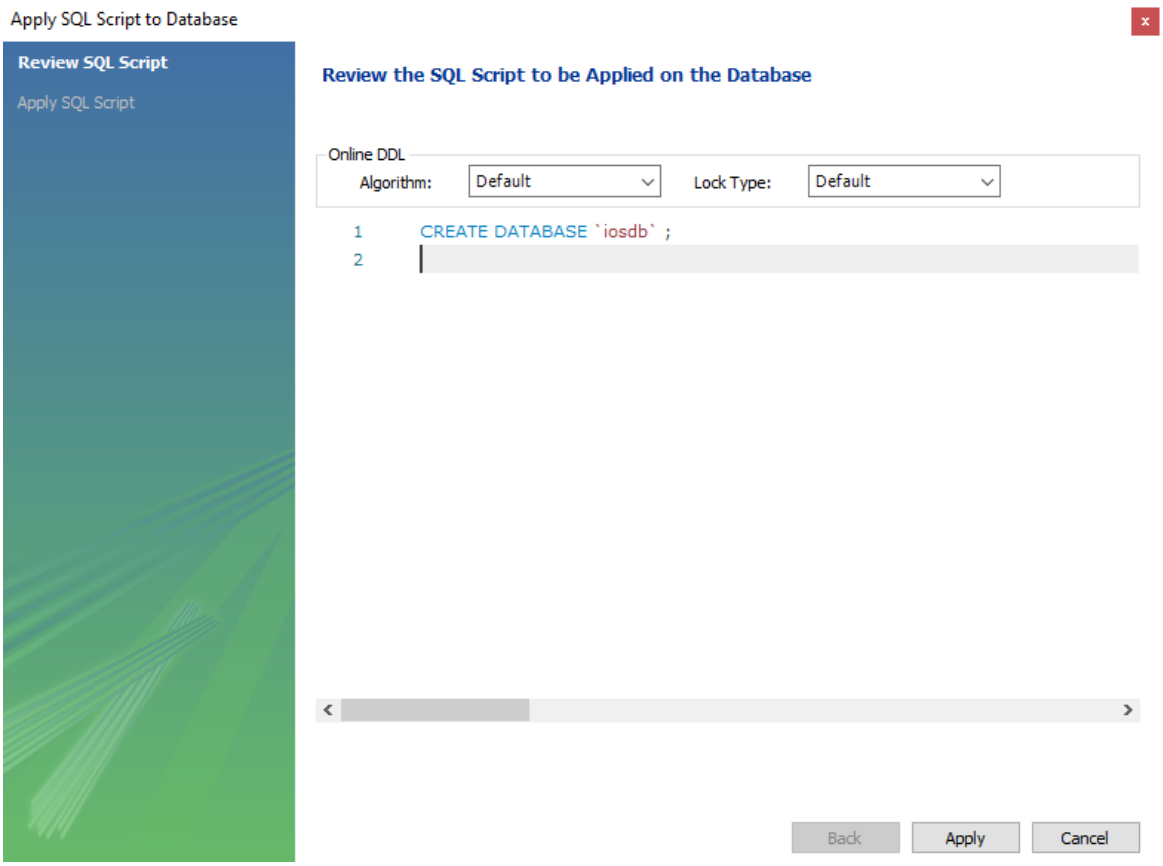


FIGURE 5.17 – Création base IOSDB

Ensuite, nous avons crée la table

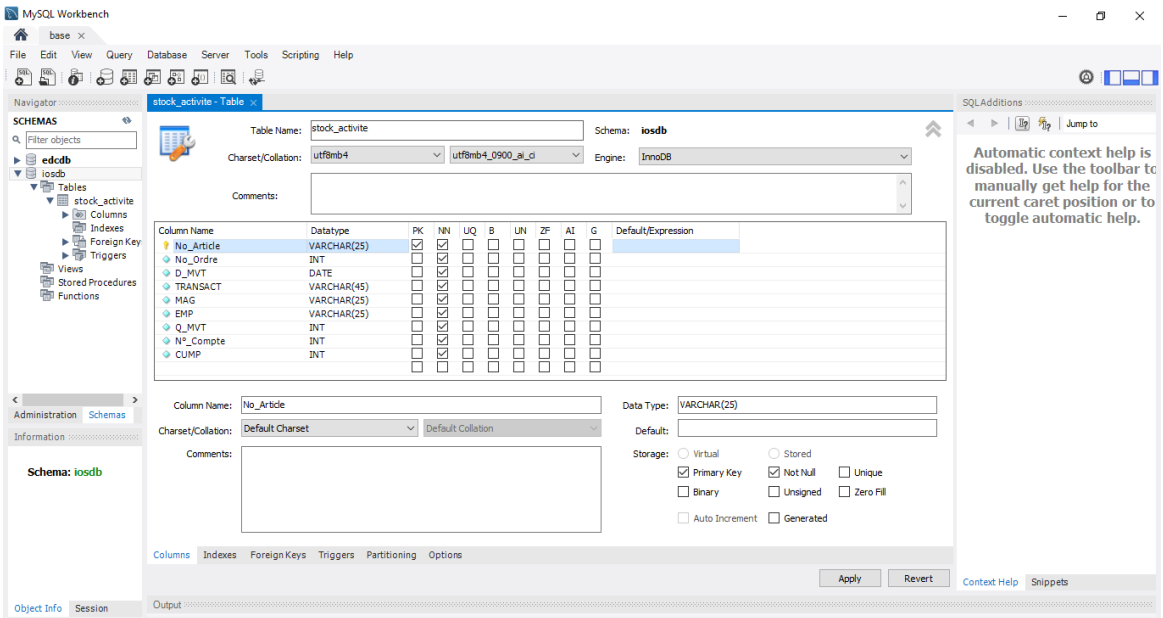


FIGURE 5.18 – Création table 4

Nous avons complétez la table et le résultat est comme suit :

| No_Article | No_Ordre | D_MVT | TRANSACT | MAG | EMP | Q_MVT | No_Compte | CUMP |
|---|---|---|---|---|---|---|---|---|
| 140819 | 863203 | 2022-03-15 | CONSOM | MI | A2C4B2 | 14 | 321010 | 129251 |
| 140819 | 863203 | 2022-02-20 | ENTREMAGASIN | MI | A2C4B2 | 25 | 321010 | 129251 |
| 200684 | 860142 | 2023-11-14 | ENTREMAGASIN | ME | B7C9A1 | 2 | 310034 | 250000 |
| 258485 | 876020 | 2020-01-15 | ENTREMAGASIN | PR | D5A3B8 | 22 | 310017 | 326948 |
| 306845 | 924820 | 2024-10-26 | CONSOM | CR | F8J4A7 | 89 | 321010 | 121547 |
| 306845 | 928420 | 2024-01-12 | ENTREMAGASIN | CR | F8J4A7 | 168 | 321010 | 121547 |
| 402132 | 916230 | 2023-04-09 | ENTREMAGASIN | MJ | E5A4D5 | 294 | 310016 | 147000 |
| 951357 | 903450 | 2022-06-18 | ENTREMAGASIN | CU | C7A3D8 | 0 | 321031 | 101547 |
| M59753 | 928206 | 2024-05-18 | CONSOM | RD | A9B4C8 | 113 | 399999 | 450123 |
| M59753 | 928206 | 2024-02-06 | ENTREMAGASIN | RD | A9B4C8 | 250 | 399999 | 450123 |
| M64821 | 923130 | 2024-10-30 | ENTREMAGASIN | FR | A9B7C2 | 67 | 399999 | 230125 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

FIGURE 5.19 – Table (STOCK-ACTIVITE)

## 3.3. Ajout d'une colonne

Après la création des tables, nous avons ajouté de nouvelles colonnes pour répondre à des besoins évolutifs du système d'information. L'ajout d'une colonne permet d'intégrer de nouvelles informations sans alerter les données existantes. Nous avons ajouté une colonne « D-MAJ » qui signifie la date de modification d'un article comme suit :

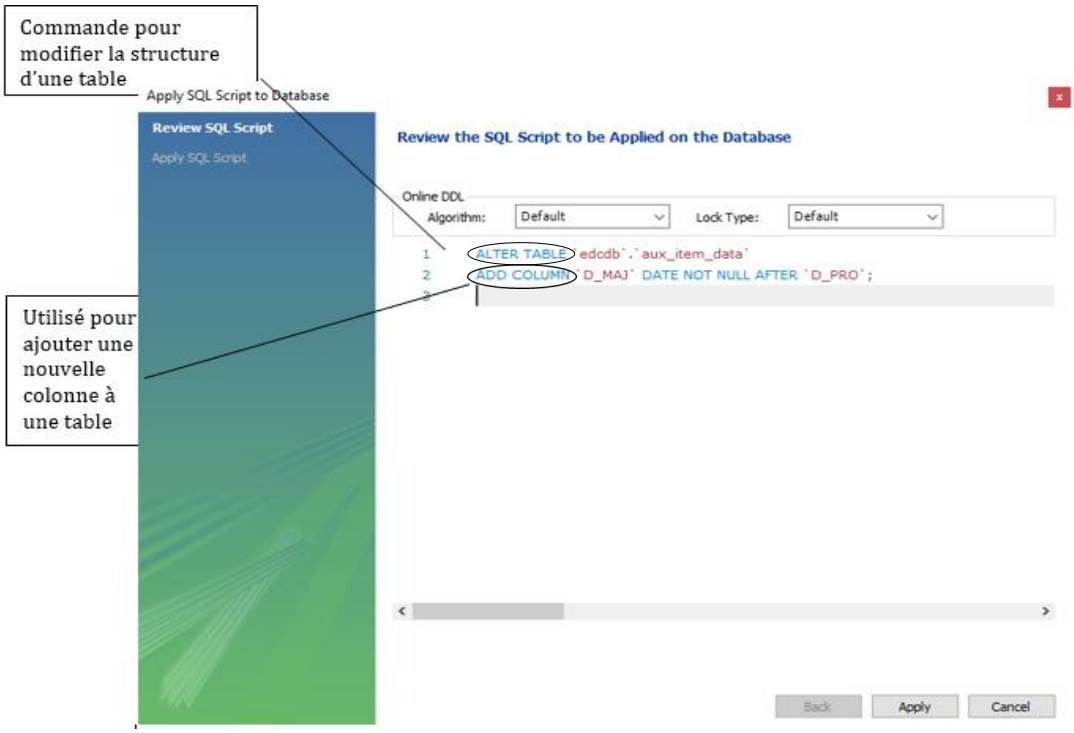


FIGURE 5.20 – Ajout d'une colonne

La table 2 (AUX-ITEM-DATA) devient comme suit :

| No_Article | No_Ordre | D_Création | D_Pro | D_MAJ | EMP | MG | ETAT | No_Compte |
|---|---|---|---|---|---|---|---|---|
| 140819 | 863203 | 2014-03-02 | 2021-06-15 | 2022-11-03 | A2C4B2 | MI | S | 321010 |
| 200684 | 860142 | 2014-12-30 | 2017-06-20 | 2020-04-14 | B7C9A1 | ME | A | 310016 |
| 258485 | 876020 | 2015-06-21 | 2018-02-14 | 2019-01-20 | D5A3B8 | PR | A | 321031 |
| 306845 | 928420 | 2020-11-01 | 2023-11-16 | 2024-05-12 | F8J4A7 | CR | N | 321010 |
| 402132 | 916230 | 2019-09-29 | 2022-09-10 | 2023-11-20 | E5A4D5 | MJ | S | 310034 |
| 951357 | 903450 | 2018-07-15 | 2021-12-22 | 2023-09-14 | C7A3D8 | CU | A | 320017 |
| M59753 | 928206 | 2020-06-27 | 2022-03-15 | 2023-04-12 | A9B4C8 | RD | S | 399999 |
| M64821 | 923130 | 2020-09-15 | 2022-06-20 | 2023-09-06 | A9B7C2 | FR | N | 399999 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

FIGURE 5.21 – Table 2 MAJ

## 3.4. Modifier nom des colonnes

Ensuite nous avons modifié les noms des colonnes pour faciliter la sélection, suivant les syntaxes suivantes, illustrée dans la figure ci-dessous :

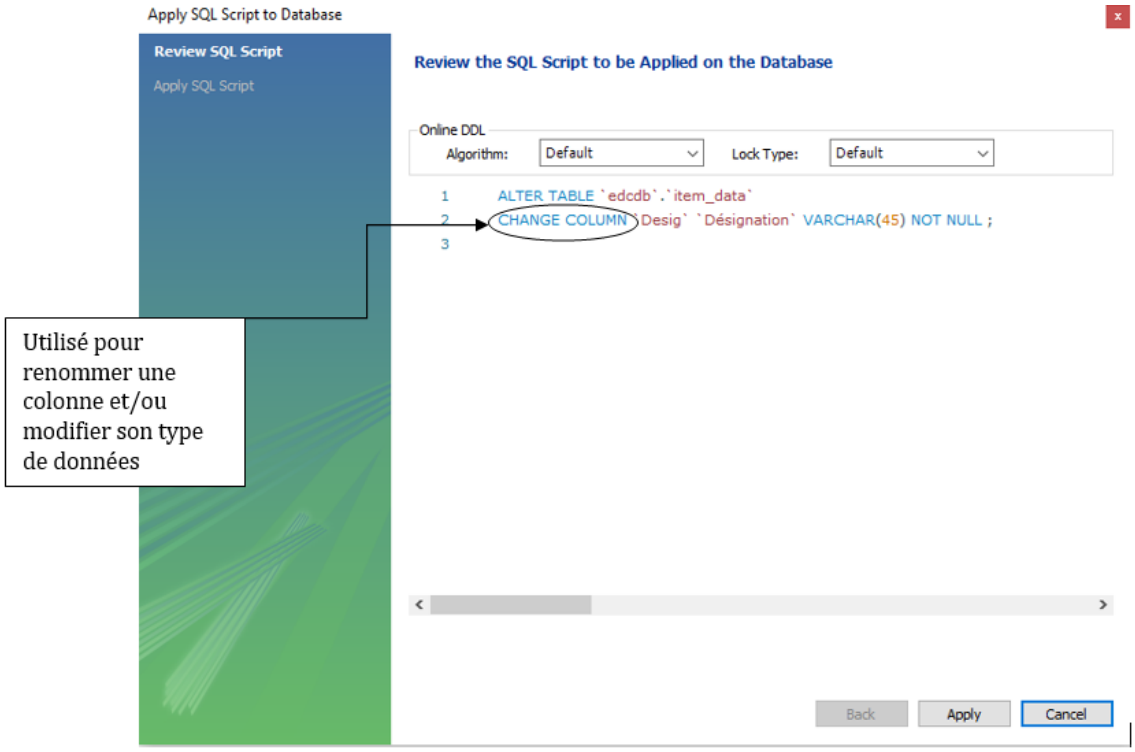


FIGURE 5.22 – Syntaxe changement Pour Table 1

La table devient comme suit :

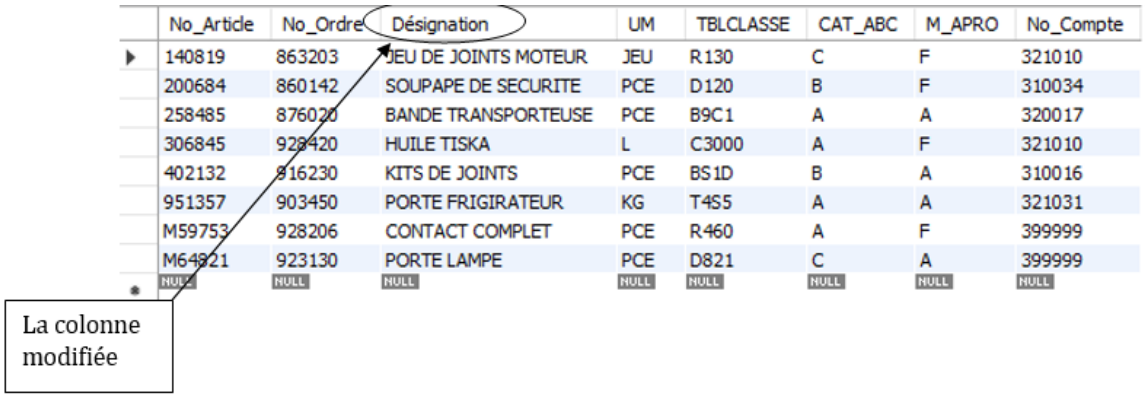


FIGURE 5.23 – Changement table 1

## 3.5. Suppression d'une colonne

Si une colonne devient obsolète ou inutile, il est possible de la supprimer afin d'optimiser la structure de la base de données et d'éviter le stockage de données superflues comme suit :

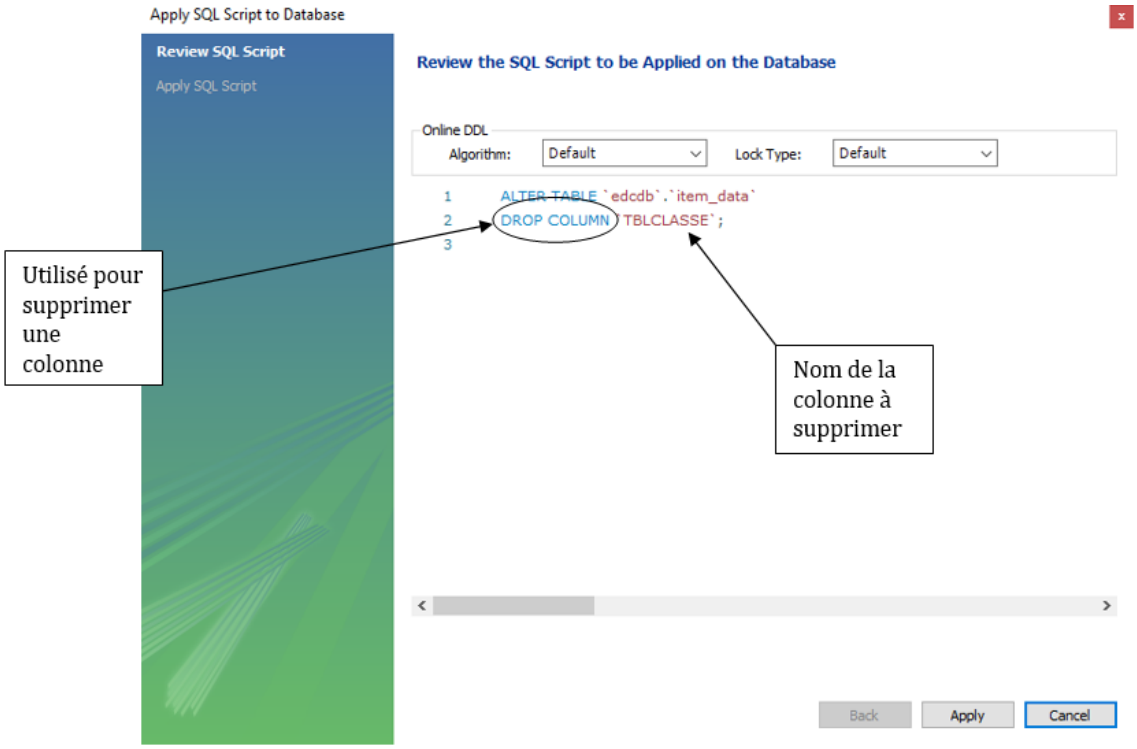


FIGURE 5.24 – Syntaxe suppression de colonne

### 3.5.1. Remarque

Cette action est irréversible, les données contenues dans la colonne seront définitivement perdues, seulement si on effectue une sauvegarde avant toute modification structurelle.
Voila les tables après les modifications :

| No_Article | No_Ordre | Desig | UM | TBLCLASSE | CAT_ABC | M_APRO | No_Compte |
|---|---|---|---|---|---|---|---|
| 140819 | 863203 | JEU DE JOINTS MOTEUR | JEU | R130 | C | F | 321010 |
| 200684 | 860142 | SOUPAPE DE SECURITE | PCE | D120 | B | F | 310034 |
| 258485 | 876020 | BANDE TRANSPORTEUSE | PCE | B9C1 | A | A | 320017 |
| 306845 | 928420 | HUILE TISKA | L | C3000 | A | F | 321010 |
| 402132 | 916230 | KITS DE JOINTS | PCE | BS1D | B | A | 310016 |
| 951357 | 903450 | PORTE FRIGIRATEUR | KG | T4S5 | A | A | 321031 |
| M59753 | 928206 | CONTACT COMPLET | PCE | R460 | A | F | 399999 |
| M64821 | 923130 | PORTE LAMPE | PCE | D821 | C | A | 399999 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

FIGURE 5.25 – Table ITEM-DATA

| No_Article | No_Ordre | D_Création | D_Pro | D_MAJ | EMP | MG | ETAT | No_Compte |
|---|---|---|---|---|---|---|---|---|
| 140819 | 863203 | 2014-03-02 | 2021-06-15 | 2022-11-03 | A2C4B2 | MI | S | 321010 |
| 200684 | 860142 | 2014-12-30 | 2017-06-20 | 2020-04-14 | B7C9A1 | ME | A | 310016 |
| 258485 | 876020 | 2015-06-21 | 2018-02-14 | 2019-01-20 | D5A3B8 | PR | A | 321031 |
| 306845 | 928420 | 2020-11-01 | 2023-11-16 | 2024-05-12 | F8J4A7 | CR | N | 321010 |
| 402132 | 916230 | 2019-09-29 | 2022-09-10 | 2023-11-20 | E5A4D5 | MJ | S | 310034 |
| 951357 | 903450 | 2018-07-15 | 2021-12-22 | 2023-09-14 | C7A3D8 | CU | A | 320017 |
| M59753 | 928206 | 2020-06-27 | 2022-03-15 | 2023-04-12 | A9B4C8 | RD | S | 399999 |
| M64821 | 923130 | 2020-09-15 | 2022-06-20 | 2023-09-06 | A9B7C2 | FR | N | 399999 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

FIGURE 5.26 – Table AUX-ITEM-DATA

| No_Article | No_Ordre | D_MVT | TRANSACT | MAG | EMP | Q_MVT | No_Compte | CUMP |
|---|---|---|---|---|---|---|---|---|
| 140819 | 863203 | 2022-03-15 | CONSOM | MI | A2C4B2 | 14 | 321010 | 129251 |
| 140819 | 863203 | 2022-02-20 | ENTREMAGASIN | MI | A2C4B2 | 25 | 321010 | 129251 |
| 200684 | 860142 | 2023-11-14 | ENTREMAGASIN | ME | B7C9A1 | 2 | 310034 | 250000 |
| 258485 | 876020 | 2020-01-15 | ENTREMAGASIN | PR | D5A3B8 | 22 | 310017 | 326948 |
| 306845 | 924820 | 2024-10-26 | CONSOM | CR | F8J4A7 | 89 | 321010 | 121547 |
| 306845 | 928420 | 2024-01-12 | ENTREMAGASIN | CR | F8J4A7 | 168 | 321010 | 121547 |
| 402132 | 916230 | 2023-04-09 | ENTREMAGASIN | MJ | E5A4D5 | 294 | 310016 | 147000 |
| 951357 | 903450 | 2022-06-18 | ENTREMAGASIN | CU | C7A3D8 | 0 | 321031 | 101547 |
| M59753 | 928206 | 2024-05-18 | CONSOM | RD | A9B4C8 | 113 | 399999 | 450123 |
| M59753 | 928206 | 2024-02-06 | ENTREMAGASIN | RD | A9B4C8 | 250 | 399999 | 450123 |
| M64821 | 923130 | 2024-10-30 | ENTREMAGASIN | FR | A9B7C2 | 67 | 399999 | 230125 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

FIGURE 5.27 – Table Stock-Activite

## 3.6. Sélection des informations sur un article

La sélection des informations sur un article permet d'extraire des données précises pour l'afficher, la gestion ou l'analyse.

— **Extraire les données d'un article** : Nous souhaitons récupérer toutes les articles dont l'unité de mesure est « PCE » (pièce). Pour cela, nous avons suivi cette l'étape afin de compléter la tache :

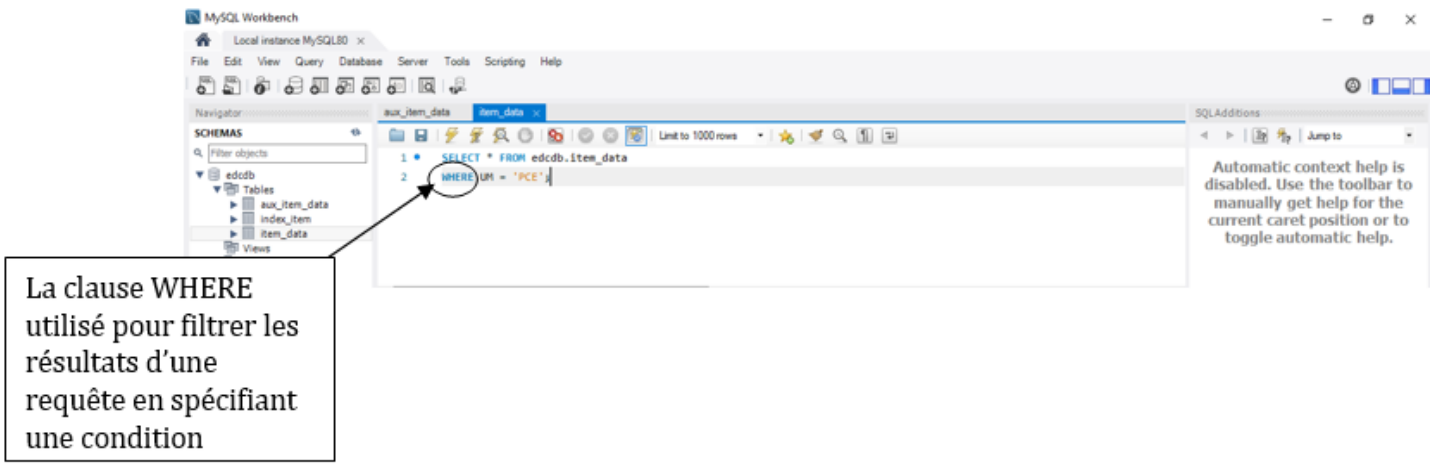


FIGURE 5.28 – Sélection UM

Résultat :

| | No_Article | No_Ordre | Désignation | UM | TBLCLASSE | CAT_ABC | M_APRO | No_Compte |
|---|---|---|---|---|---|---|---|---|
| ▶ | 200684 | 860142 | SOUPAPE DE SECURITE | PCE | D120 | B | F | 310034 |
| | 258485 | 876020 | BANDE TRANSPORTEUSE | PCE | B9C1 | A | A | 320017 |
| | 402132 | 916230 | KITS DE JOINTS | PCE | BS1D | B | A | 310016 |
| | M59753 | 928206 | CONTACT COMPLET | PCE | R460 | A | F | 399999 |
| | M64821 | 923130 | PORTE LAMPE | PCE | D821 | C | A | 399999 |
| * | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

FIGURE 5.29 – Table de condition UM

— Ensuite nous avons effectué une jointure pour sélectionner quelques informations à partir des deux tables simultanément comme suit :

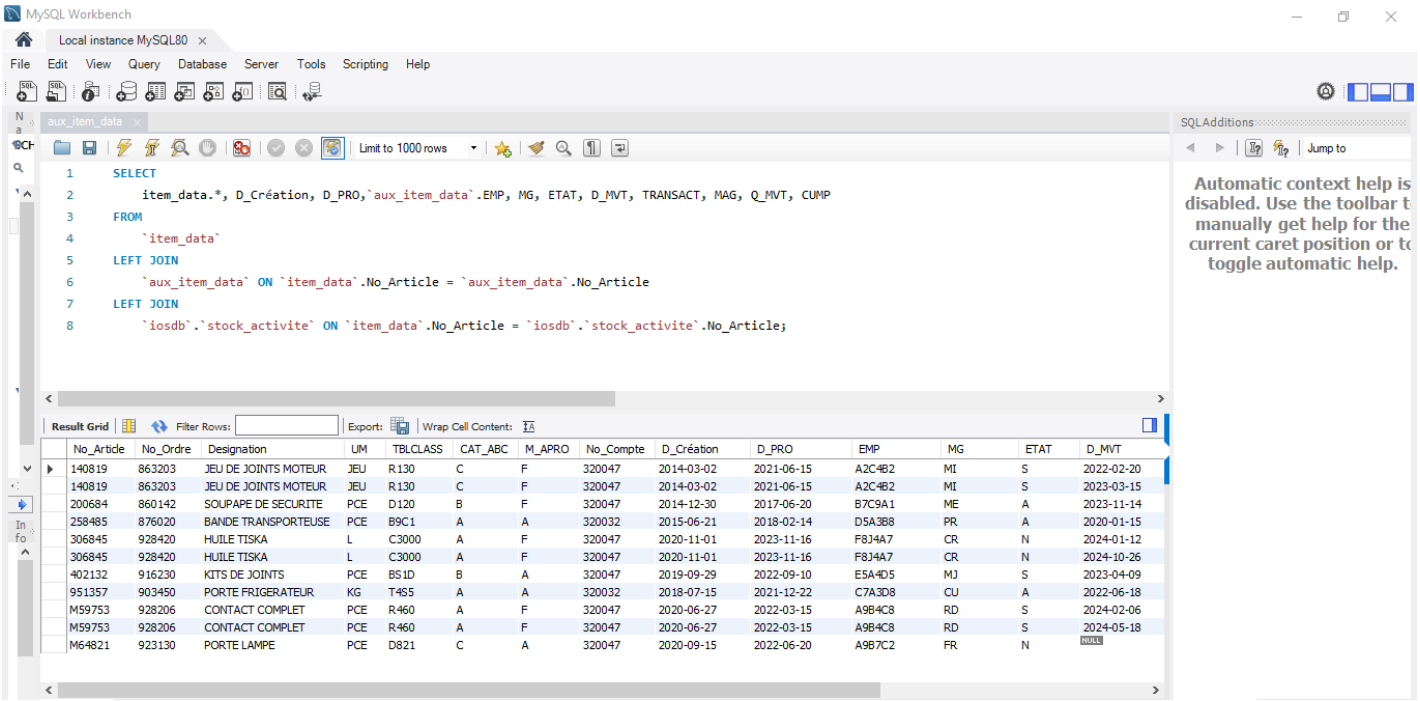


FIGURE 5.30 – Syntaxe jointure

Dans ce cas on a utilisé :

— la commande INNER JOIN qui permet de récupérer des données communes à deux tables.

— La clause ON pour spécifier la condition qui lie les deux tables.

Le résultat obtenu est comme suit :

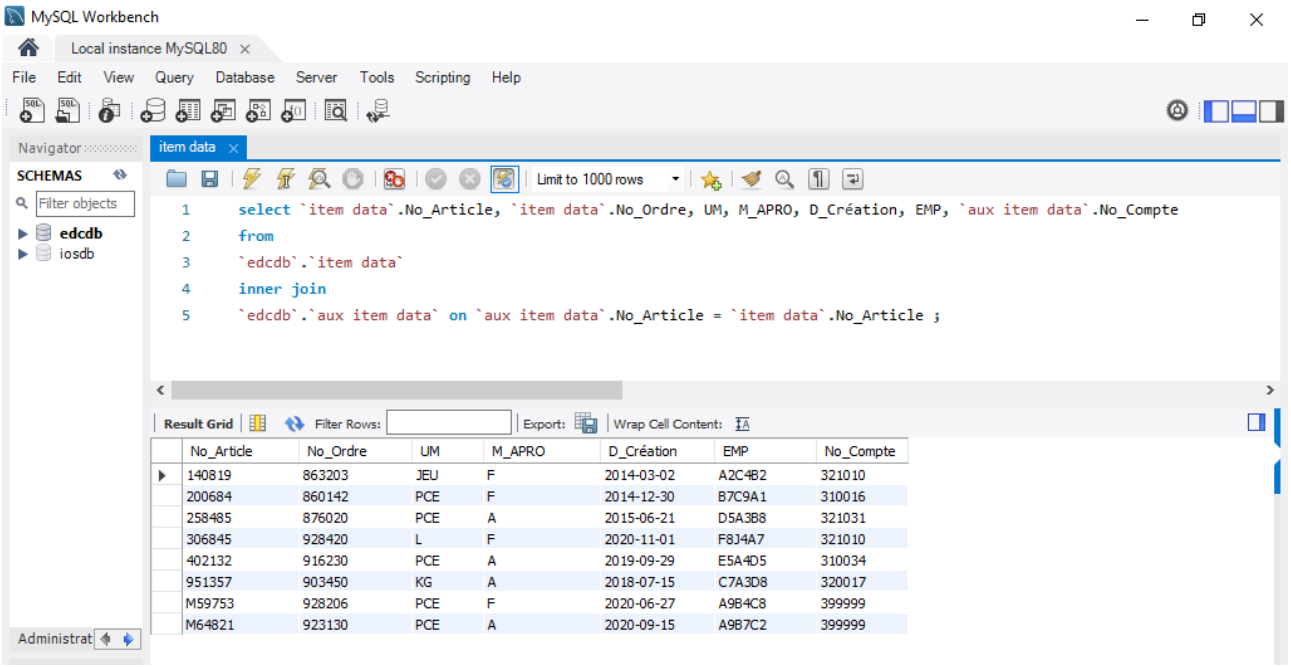


FIGURE 5.31 – Table Résultat jointure

— Ensuite nous avons effectué une jointure pour sélectionner toutes les informations essentielles à partir des trois tables simultanément comme suit :

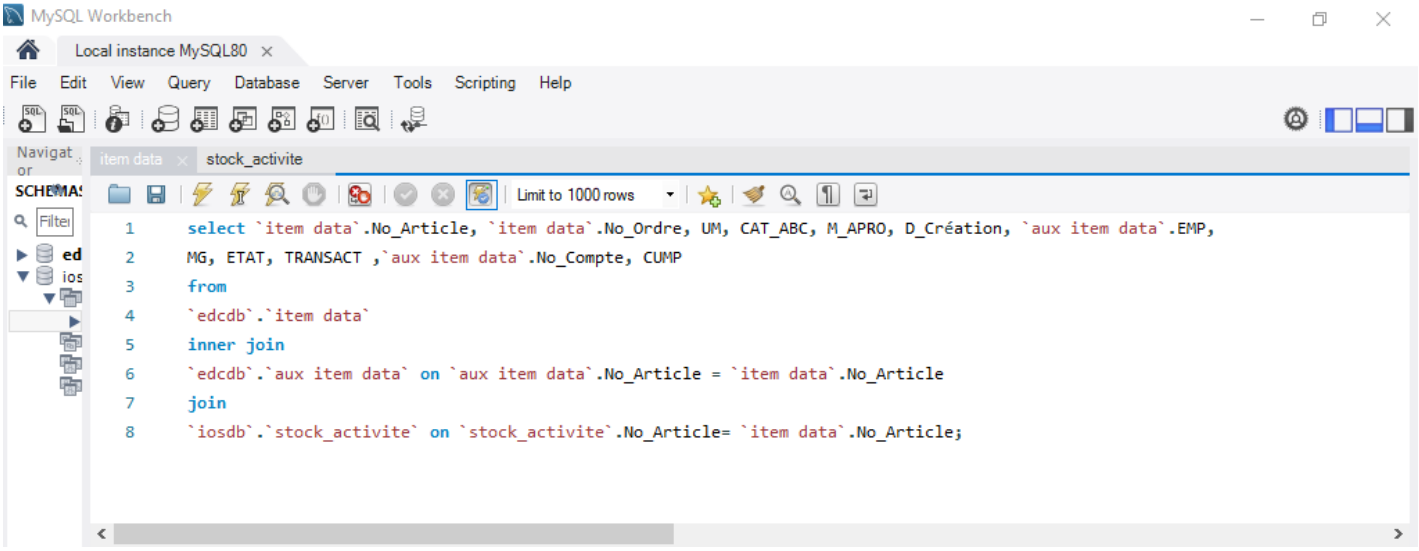


FIGURE 5.32 – Syntaxe jointure 2

Le résultat obtenu comme suit :

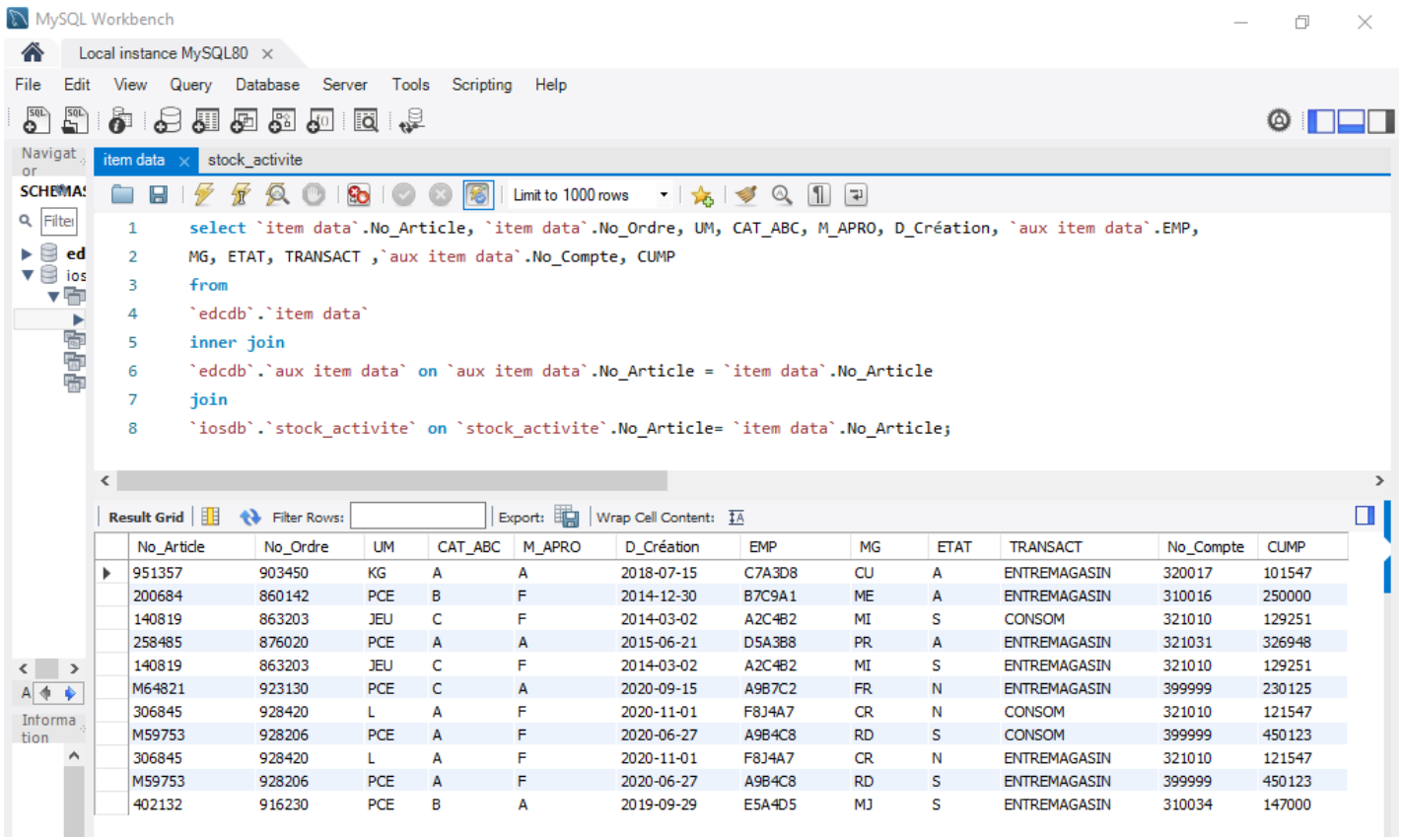


FIGURE 5.33 – Table jointure 2

## 3.7. Résolution de la problématique

Avant d'entamer la résolution de la problématique, nous avons appliqué la syntaxe suivante afin de détecter d'éventuelles erreurs sur la colonne « No-Compte » :

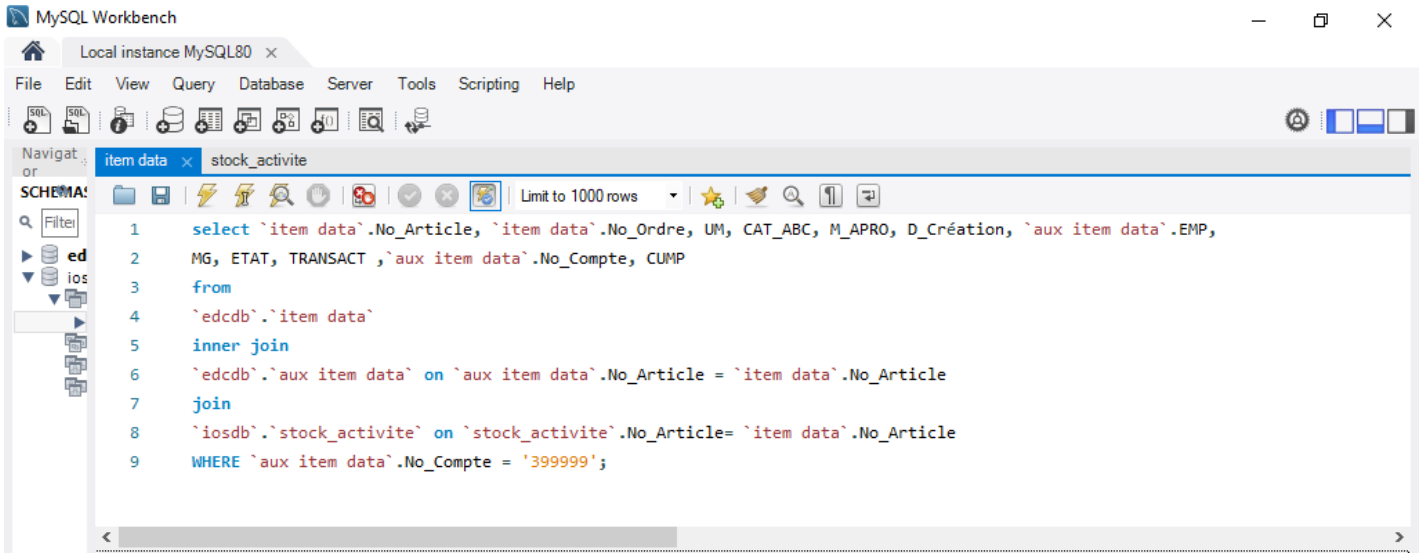


FIGURE 5.34 – Syntaxe Problématique

Le résultat affiche comme suit :

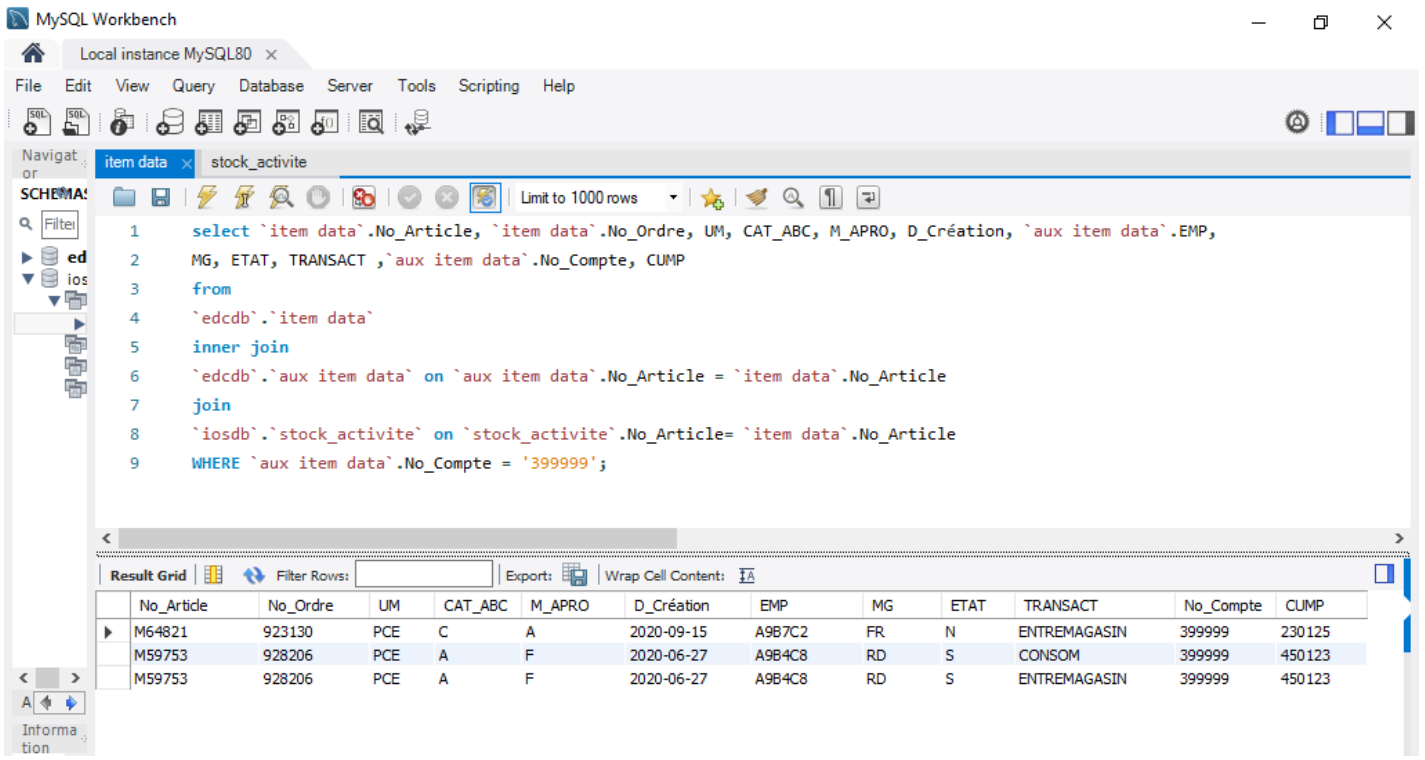


FIGURE 5.35 – Table Problématique.

Une fois l'erreur est détectée, nous avons transmis les numéros des articles concernés au service Finance et Comptabilité afin qu'ils puissent déterminer les comptes comptables correspondants. Une fois les informations reçues, nous avons procédé à la correction nécessaire et relancé le transfert des journaux vers le système EASY.

Nous avons modifié les numéros comme suit pour chaque table : Pour la table 1 (ITEM-DATA) :

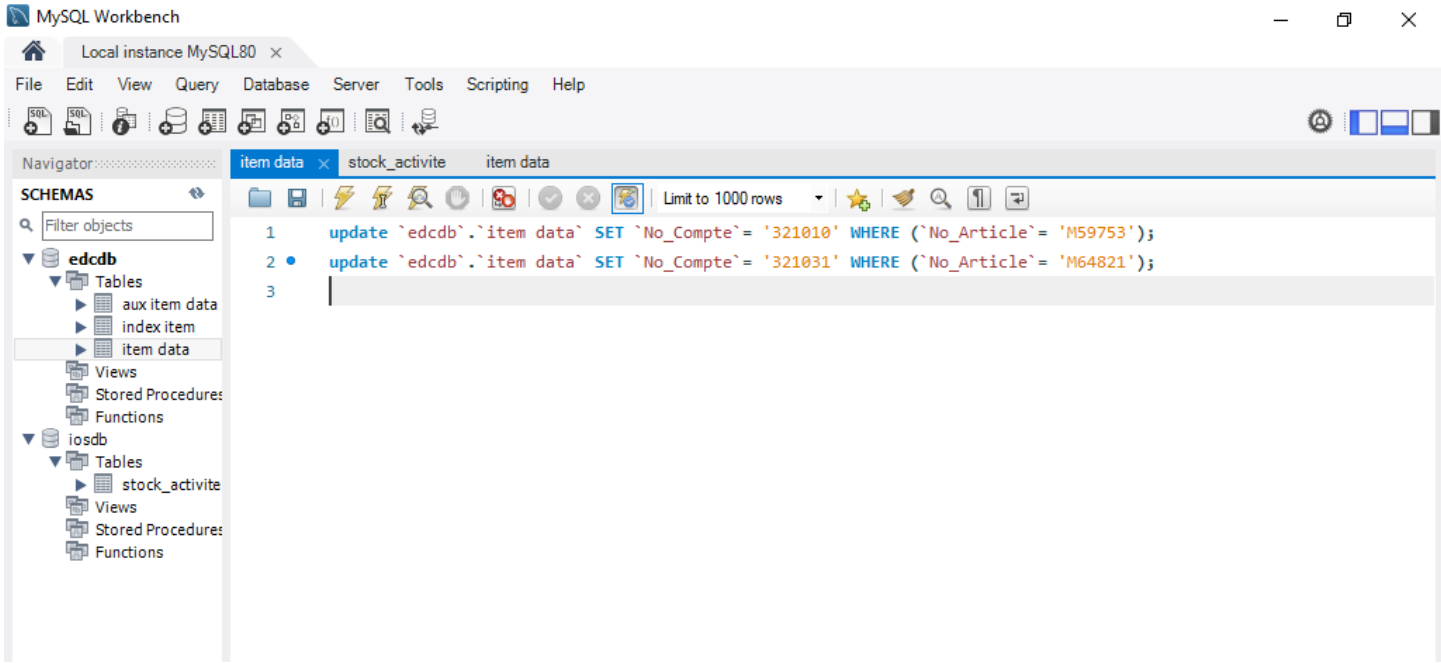


FIGURE 5.36 – Mise à jour pour la Table 1

Pour la Table 2 (AUX-ITEM-DATA) :

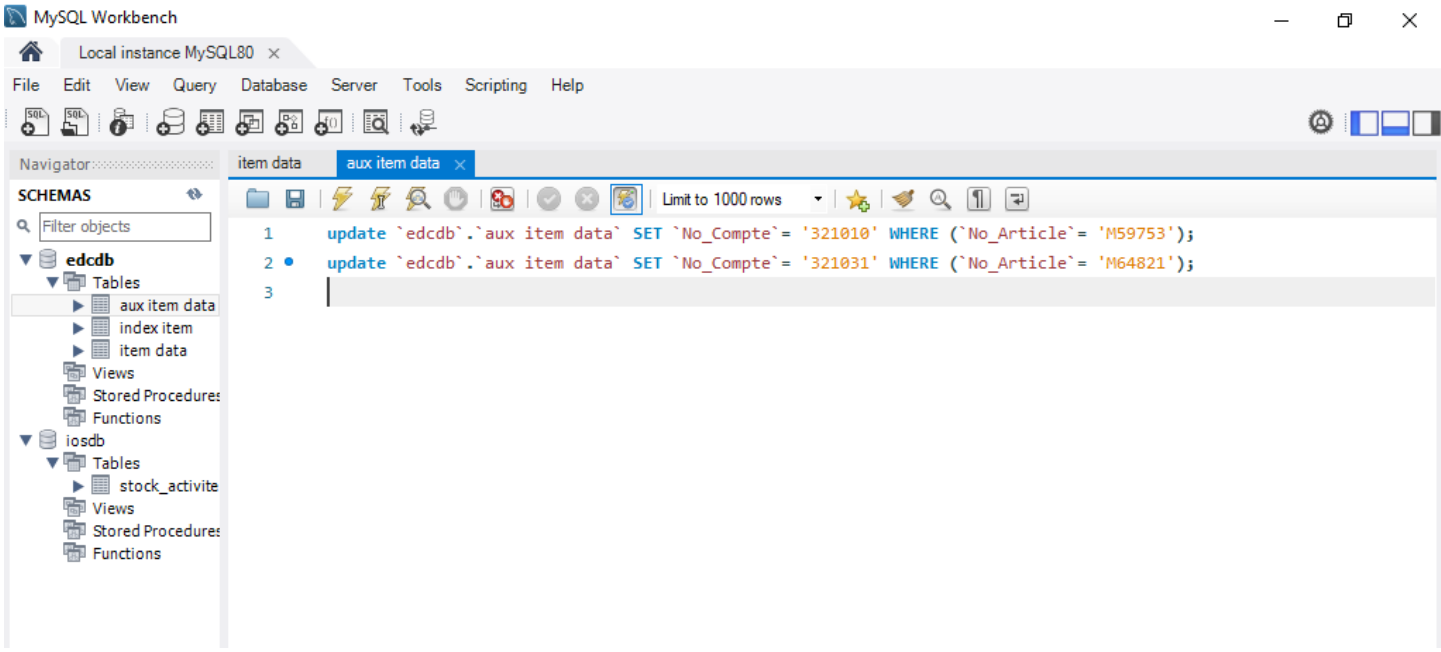


FIGURE 5.37 – Mise à jour pour la table 2

Pour la table (STOCK-ACTIVITE) :

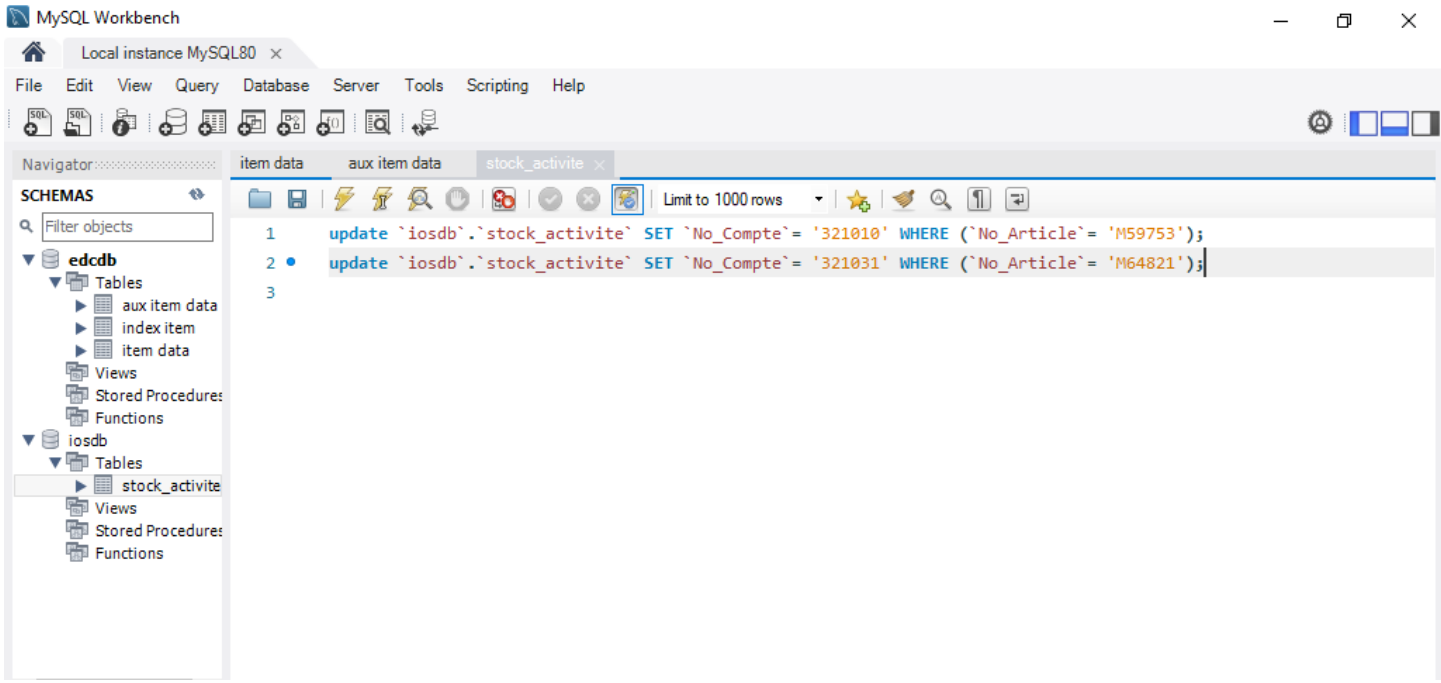


FIGURE 5.38 – Mise à jour pour la table 4

Enfin, le problème a été résolu pour la confirmation on effectué cette opération :

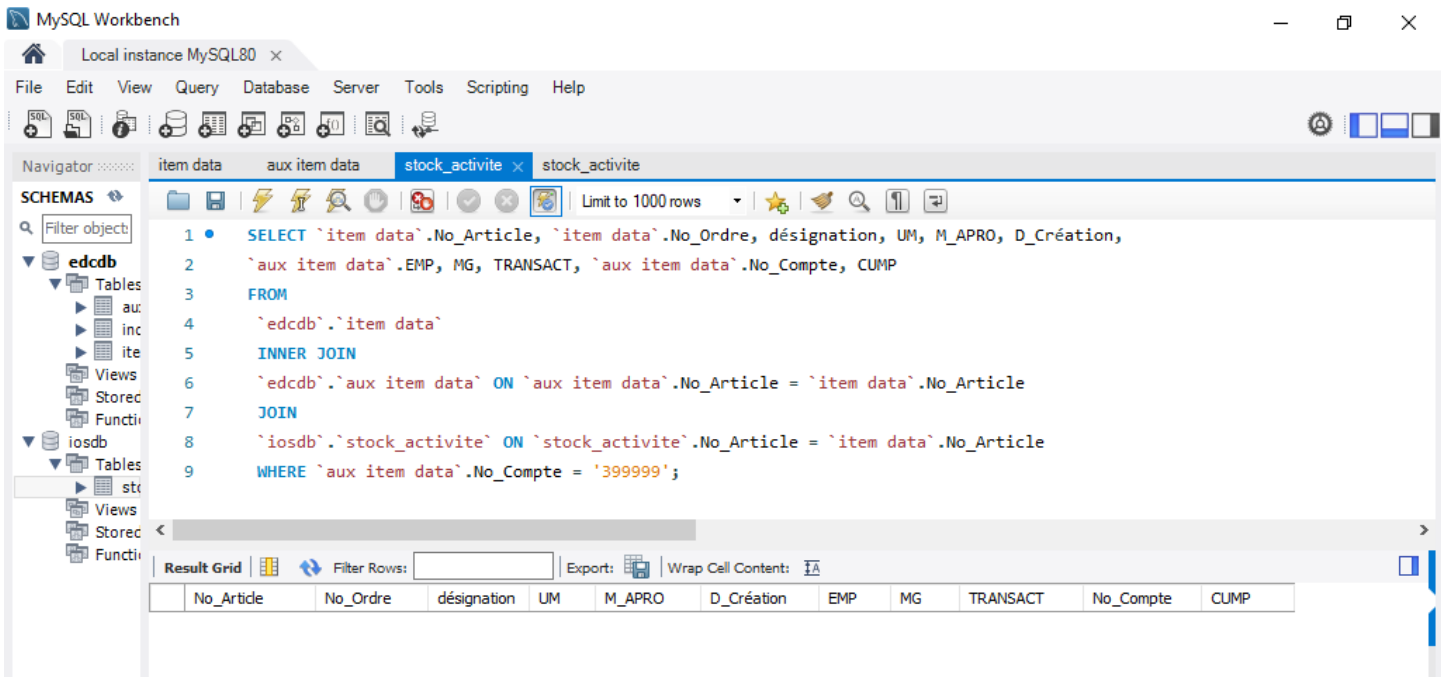


FIGURE 5.39 – Résolution du problème

Nous avons reformulé la jointure pour vérifier et sélectionner toutes les informations essentielles à partir des trois tables simultanément comme suit :

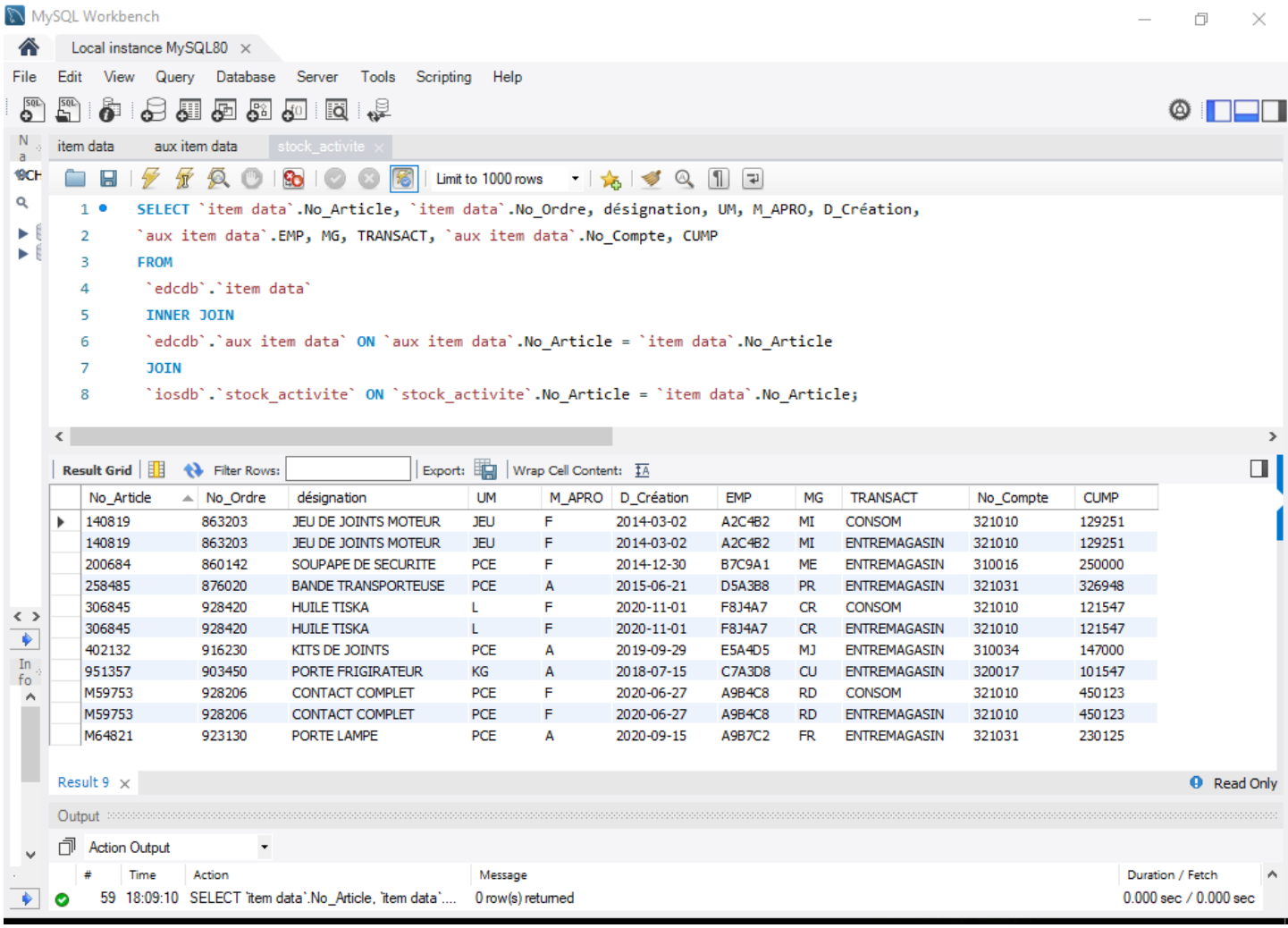


FIGURE 5.40 – Vérification

# 4. Problématique 2

Lors de l'analyse de la structure de la base de données en cours de développement, nous avons constaté une redondance notable des données dans trois tables : « item-data », « aux-item-data » et « stock-activity ». Plus précisément, les colonnes ''No-Compte" et ''No-Ordre" sont répétées dans chacune de ces tables, alors qu'elles ne représentent ni des clés primaires, ni des identifiants uniques. Cette duplication engendre plusieurs inconvénients :

— Une augmentation inutile de la taille de la base de données.
— Un risque de non-cohérence entre les tables en cas de mise à jour manuelle.
— Une complexification des requêtes SQL (notamment pour les jointures et les filtrages).

## 4.1. Proposition de résolution

Afin d'optimiser la structure de la base de données, il est proposé de :

— Supprimer les colonnes ''No-Compte" et ''No-Ordre" des tables « aux-item-data » et « stock-activity ».

— Conserver ces informations uniquement dans la table « item-data », considérée comme table de référence.
— Mettre en place des jointures appropriées (via une clé étrangère) pour récupérer ces données au besoin depuis « item-data ».

Cette approche permet :

— De respecter les principes de normalisation de bases de données.
— De réduire la redondance et améliorer l'intégrité des données.
— De faciliter la maintenance et l'évolution de la base à long terme.

## 4.2. Déroulement de la résolution

Premièrement on va supprimer les 2 colonnes ''No-Ordre" et ''No-Compte " dans la table « aux-item-data » :

| No_Article | No_Ordre | D_Création | D_PRO | EMP | MG | ETAT | No_Compte |
|---|---|---|---|---|---|---|---|
| 140819 | 863203 | 2014-03-02 | 2021-06-15 | A2C4B2 | MI | S | 320047 |
| 200684 | 860142 | 2014-12-30 | 2017-06-20 | B7C9A1 | ME | A | 320047 |
| 258485 | 876020 | 2015-06-21 | 2018-02-14 | D5A3B8 | PR | A | 320032 |
| 306845 | 928420 | 2020-11-01 | 2023-11-16 | F8J4A7 | CR | N | 320047 |
| 402132 | 916230 | 2019-09-29 | 2022-09-10 | E5A4D5 | MJ | S | 320047 |
| 951357 | 903450 | 2018-07-15 | 2021-12-22 | C7A3D8 | CU | A | 320032 |
| M59753 | 928206 | 2020-06-27 | 2022-03-15 | A9B4C8 | RD | S | 320047 |
| M64821 | 923130 | 2020-09-15 | 2022-06-20 | A9B7C2 | FR | N | 320047 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

FIGURE 5.41 – table problématique

On applique la syntaxe suivante pour les supprimer :

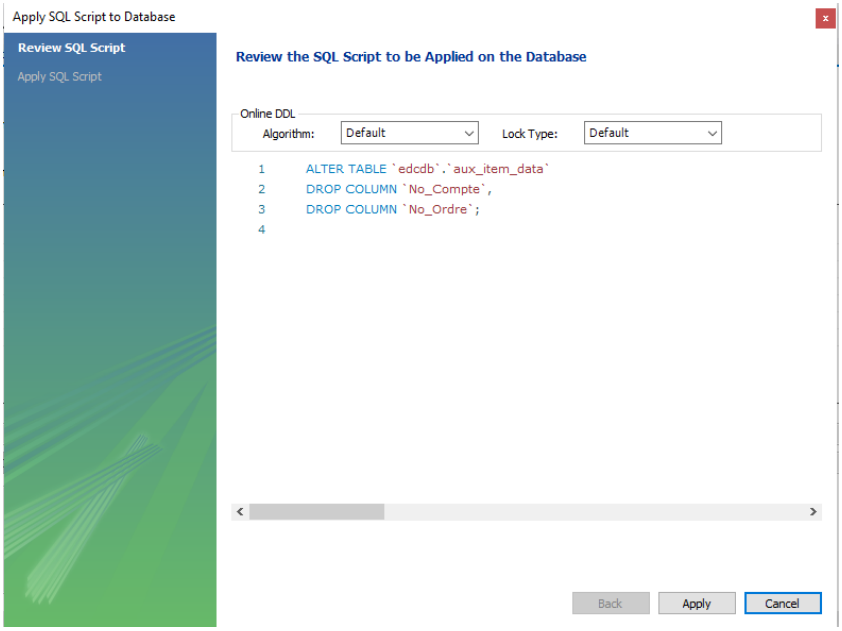


FIGURE 5.42 – Syntaxe supprimer

On aura le résultat :

| No_Article | D_Création | D_PRO | EMP | MG | ETAT |
|---|---|---|---|---|---|
| 140819 | 2014-03-02 | 2021-06-15 | A2C4B2 | MI | S |
| 200684 | 2014-12-30 | 2017-06-20 | B7C9A1 | ME | A |
| 258485 | 2015-06-21 | 2018-02-14 | D5A3B8 | PR | A |
| 306845 | 2020-11-01 | 2023-11-16 | F8J4A7 | CR | N |
| 402132 | 2019-09-29 | 2022-09-10 | E5A4D5 | MJ | S |
| 951357 | 2018-07-15 | 2021-12-22 | C7A3D8 | CU | A |
| M59753 | 2020-06-27 | 2022-03-15 | A9B4C8 | RD | S |
| M64821 | 2020-09-15 | 2022-06-20 | A9B7C2 | FR | N |
| NULL | NULL | NULL | NULL | NULL | NULL |

FIGURE 5.43 – Résultat

Après on supprime les mêmes colonnes dans la table « stock-activity »

| No_Article | No_Ordre | D_MVT | TRANSACT | MAG | EMP | Q_MVT | No_Compte | CUMP |
|---|---|---|---|---|---|---|---|---|
| 140819 | 863203 | 2023-03-15 | CONSOM | MI | A2C4B2 | 14 | 320047 | 129251 |
| 140819 | 863203 | 2022-02-20 | ENTERMAGASIN | MI | A2C4B2 | 25 | 320047 | 189251 |
| 200684 | 860142 | 2023-11-14 | ENTERMAGASIN | ME | B7C9A1 | 2 | 320047 | 250000 |
| 258485 | 876020 | 2020-01-15 | ENTERMAGASIN | PR | D5A3B8 | 22 | 320032 | 326948 |
| 306845 | 924820 | 2024-10-26 | CONSOM | CR | F8J4A7 | 89 | 320047 | 121547 |
| 306845 | 924820 | 2024-01-12 | ENTERMAGASIN | CR | F8J4A7 | 168 | 320047 | 121547 |
| 402132 | 916230 | 2023-04-09 | ENTERMAGASIN | MJ | E5A4D5 | 294 | 320047 | 147000 |
| 951357 | 903450 | 2022-06-18 | ENTERMAGASIN | CU | C7A3D8 | 0 | 320032 | 101547 |
| M59753 | 928206 | 2024-05-18 | CONSOM | RD | A9B4C8 | 113 | 320047 | 450123 |
| M59753 | 928206 | 2024-02-06 | ENTERMAGASIN | RD | A9B4C8 | 250 | 399999 | 450123 |

FIGURE 5.44 – stock-activité

En suivant cette étape de la suppression avec la syntaxe représenter dans cette figure :

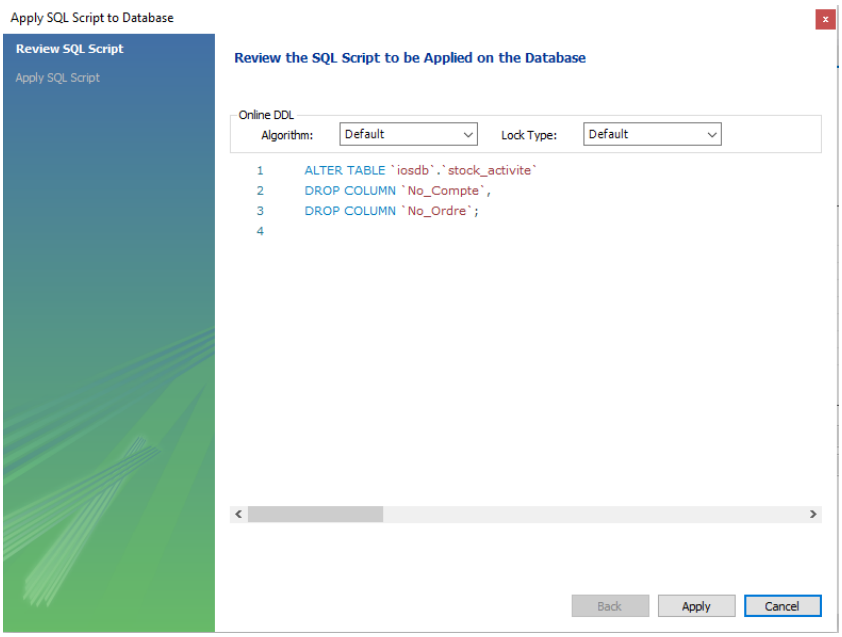


FIGURE 5.45 – syntaxe stock-activité

On aura ce résultat :

| No_Article | D_MVT | TRANSACT | MAG | EMP | Q_MVT | CUMP |
|---|---|---|---|---|---|---|
| 140819 | 2023-03-15 | CONSOM | MI | A2C4B2 | 14 | 129251 |
| 140819 | 2022-02-20 | ENTERMAGASIN | MI | A2C4B2 | 25 | 189251 |
| 200684 | 2023-11-14 | ENTERMAGASIN | ME | B7C9A1 | 2 | 250000 |
| 258485 | 2020-01-15 | ENTERMAGASIN | PR | D5A3B8 | 22 | 326948 |
| 306845 | 2024-10-26 | CONSOM | CR | F8J4A7 | 89 | 121547 |
| 306845 | 2024-01-12 | ENTERMAGASIN | CR | F8J4A7 | 168 | 121547 |
| 402132 | 2023-04-09 | ENTERMAGASIN | MJ | E5A4D5 | 294 | 147000 |
| 951357 | 2022-06-18 | ENTERMAGASIN | CU | C7A3D8 | 0 | 101547 |
| M59753 | 2024-05-18 | CONSOM | RD | A9B4C8 | 113 | 450123 |
| M59753 | 2024-02-06 | ENTERMAGASIN | RD | A9B4C8 | 250 | 450123 |

FIGURE 5.46 – resultat

Maintenant pour avoir toutes les informations, on a appliqué la jointure pour avoir une table bien structurée contient toutes les informations sans la répétition :

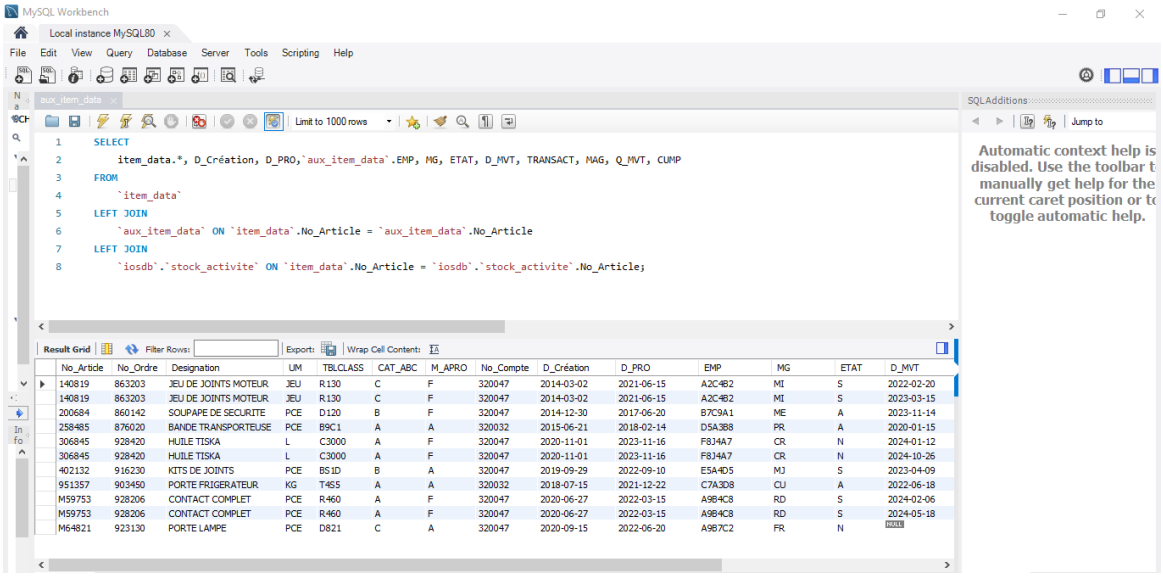


FIGURE 5.47 – jointure

# Conclusion Générale

Le langage SQL (Structured Query Language) occupe une place centrale dans le domaine de la gestion des bases de données. Il est utilisé de manière universelle pour créer, manipuler, interroger et sécuriser les données au sein de systèmes d'information variés. Grâce à sa puissance, sa simplicité syntaxique et sa standardisation, SQL est devenu un outil incontournable dans presque tous les secteurs d'activité où la gestion de données est cruciale, comme la santé, la finance, l'éducation ou encore l'industrie.

Ce mémoire a eu pour objectif principal d'explorer en profondeur le fonctionnement et l'utilisation du langage SQL, en partant des bases théoriques jusqu'à son application pratique. Nous avons étudié les principales commandes SQL, les types de requêtes, les opérateurs, ainsi que l'environnement MySQL, l'un des systèmes de gestion de bases de données les plus populaires. Cette exploration a permis de comprendre non seulement la syntaxe du langage, mais aussi les bonnes pratiques pour structurer, interroger et exploiter efficacement une base de données.

Au-delà de l'aspect théorique, ce travail s'est inscrit dans une dynamique professionnelle à travers un stage pratique durant lequel j'ai appliqué mes connaissances pour résoudre une problématique réelle à l'aide de MySQL. Cette expérience m'a permis de consolider mes acquis et de développer une approche plus concrète de la gestion de données. Enfin, le projet de création d'une plateforme de gestion de bases de données destinée aux entreprises telles que les garages, les sociétés de bâtiment, ou les dépôts de boissons s'inscrit dans une continuité logique de ce mémoire, montrant que les compétences acquises sont utiles et peuvent être mises en pratique dans un projet d'entreprise innovant.

# Bibliographie


[1] Base-de-donnees.com. Identifiant dans une base de données.

[2] E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 1970.

[3] Oracle Corporation. *SQL Data Types*, 2023.

[4] Oracle Corporation. *SQL UNION ALL Operator*, 2023.

[5] C. J. Date. *Database Design and Relational Theory : Normal Forms and All That Jazz.* O'Reilly Media, 2012.

[6] C. J. Date. *SQL and Relational Theory : How to Write Accurate SQL Code.* O'Reilly Media, 2019.

[7] C.J. Date. *An Introduction to Database Systems.* Addison-Wesley, 2003.

[8] Ramez Elmasri and Shamkant B. Navathe. *Fundamentals of Database Systems.* Pearson, 2016.

[9] Microsoft Corporation. Select (transact-sql), 2022.

[10] MySQL. Mysql documentation.

[11] Oracle. *Database Concepts.* Oracle Corporation, 2021.

[12] Oracle Corporation. *Aggregate Functions*, 2023.

[13] Oracle Corporation. Sql conditions – logical operators, 2023.

[14] Oracle Corporation. Sql expressions - arithmetic operators, 2023.

[15] Oracle Corporation. *MySQL 8.0 Reference Manual*, 2024.

[16] Oracle Corporation. *MySQL 8.4 Reference Manual*, 2024.

[17] Oracle Corporation and Microsoft Corporation. Principes fondamentaux issus des documentations techniques oracle (2023) et microsoft sql server (2022), adaptés aux pratiques courantes en développement, 2023.

[18] Abraham Silberschatz, Henry F. Korth, and S. Sudarshan. *Database System Concepts.* McGraw-Hill Education, 2020.

[19] The PostgreSQL Global Development Group. Postgresql documentation : Comparison functions and operators, 2024.

[20] The PostgreSQL Global Development Group. Postgresql documentation : Queries, 2024.

[21] The PostgreSQL Global Development Group. Postgresql documentation : Sql queries, 2024.