Partie V :

LANGAGE SQL (RAPPELS)

- ✓ Utiliser les types de données appropriés pour chaque situation ;
- ✓ Installer et configurer un SGBD ;
- ✓ Écrire des requêtes simples à l'aide du langage SQL.

Scénario d'appui

La Pharmacie La Santé est une chaîne aui gère des médicaments provenant de divers fournisseurs, qu'elle met au service des clients. Les clients de la pharmacie sont identifiés par un identifiant unique, un nom, un prénom, une adresse, un numéro de téléphone et une date de naissance. Les médicaments sont identifiés par un identifiant unique, un nom commercial, un nom générique, une date de péremption, un prix et une catégorie (par exemple, antibiotique, analgésique) définie par un code unique et un nom. Les fournisseurs approvisionnent la pharmacie en médicaments et sont identifiés par un identifiant unique, un nom, une adresse, un numéro de téléphone. Un médicament peut être fourni par plusieurs fournisseurs et chaque fournisseur peut fournir plusieurs médicaments. Une commande peut être passée par un client et est identifiée par un identifiant unique, la date de la commande, et le statut (par exemple, en attente, expédiée, livrée). Les détails de chaque commande incluent les références des médicaments commandés, les quantités et le prix unitaire.

Scénario d'appui

Soit la base de données « pharmacie » ci-dessous représentant une implémentation de la situation décrite.

Table **clients**

ID_client	nom_client	prenom_client	adresse	telephone	date_naissance
1	Nguemeta	Paul	Rue de la Santé, Yaoundé	679000001	1985-05-12
2	Tchako	Marie	Avenue de la Paix, Douala	679000002	1990-11-03
3	Ngongang	Jean	Quartier Bafoussam	679000003	1980-08-19
4	Tita	Sylvie	Rue de l'Hôpital, Garoua	679000004	1995-02-27
5	Mvondo	Pierre	Boulevard de la Réunification, Yaoundé	679000005	1988-03-15

Table **categories**

ID_categorie	nom_categorie
1	Analgésique
2	Antibiotique
3	Anti-inflammatoire
4	Antipyrétique
5	Antihistaminique

Scénario d'appui

Table **medicaments**

ID_medicament	nom_commercial	nom_generique	ID_categorie	date_peremption	prix
1	Doliprane	Paracetamol	1	2023-12-31	500
2	Amoxicilline	Amoxicilline	2	2024-06-30	1500
3	Aspirine	Acide acétylsalicylique	1	2023-11-30	400
4	Ibuprofène	Ibuprofène	3	2024-01-15	800
5	Augmentin	Amoxicilline/Clavulanate	2	2024-05-20	2500

Table **fournisseurs**

ID_fournisseur	nom_fournisseur	adresse	telephone
1	PharmaDis	Rue de l'Industrie, Douala	679000003
2	MedSupply	Boulevard de la Santé, Yaoundé	679000004
3	DistribMed	Avenue des Pharmacies, Douala	679000005
4	HealthPharma	Rue des Hôpitaux, Bafoussam	679000006
5	MediCameroon	Quartier de la Santé, Garoua	679000007

Scénario d'appui

Table **commandes**

ID_commande	date_commande	statut	ID_client
1	2023-10-15	Livrée	1
2	2023-11-01	En attente	2
3	2023-11-20	Expédiée	3
4	2023-11-22	Livrée	4
5	2023-11-25	En attente	5

Table **fournisseur_medicament**

ID_fournisseur	ID_medicament
1	1
1	2
2	3
2	4
3	5
3	1
4	2
5	3

Table **details_commande**

ID_commande	ID_medicament	quantite	prix_unitaire
1	1	2	500
1	2	1	1500
2	3	3	400
2	1	1	500
3	4	1	800
3	5	2	2500
4	1	5	500
4	3	2	400
5	4	3	800
5	2	1	1500

I. Choix des outils et technologies

1. Langage SQL

Le choix de SQL est judicieux lorsqu'il s'agit de manipuler et gérer des bases de données relationnelles pour les quelques raisons suivantes :

- **Langage standardisé** : SQL est reconnu par des organismes de normalisation tels que l'ISO et l'ANSI. Ce sui fait qu'il est largement utilisé et utilisé dans le monde entier.
- **Familiarité avec le langage** : L'apprenant possède à ce niveau des bases sur le langage, donc la prise en main est déjà un acquis.

I. Choix des outils et technologies

1. Langage SQL

- **Compatibilité** : Il est compatible avec la plupart des SGBDR.
- **Facilité d'apprentissage** : La syntaxe est assez simple et ses commandes sont très intuitives.
- **Performance et optimisation** : Des algorithmes sophistiqués sont utilisés pour optimiser les requêtes SQL.
- **Utilisation en milieu professionnel** : Il est largement utilisé dans les entreprises pour la manipulation des données, les rapports et analyses. Les compétences en SQL sont donc très demandées sur le marché du travail.

I. Choix des outils et technologies

2. SGBD : MySQL

- **Popularité** : C'est un SGBDR largement utilisé et sert de point de base pour l'appréhension de la gestion des bases de données.
- **Open source** : Il est accessible gratuitement.
- **Compatibilité** : Il s'intègre facilement avec de nombreux langages de programmation.
- **Performance** : Il est assez efficace dans la gestion des bases de données pour des projets que l'apprenant pourra gérer dans un début d'intégration dans le monde professionnel.

II. Le langage SQL

3. Interface d'administration : phpMyAdmin

- **Interface graphique conviviale** : Dans le but de faire fi de la prise en main des commandes en ligne de commande, il facilite la gestion des bases de données MySQL (et MariaDB).
- **Accès Web** : Il est intégré dans des distributions telles que XAMPP, WAMP, MAMP (que l'apprenant utilise déjà pour des cours de programmation), il est accessible via un navigateur et peut donc facilement être utilisé en réseau pour des TPs par exemple.

II. Les types de données et contraintes d'intégrité sur MySQL

L'on sait qu'une base de données est constituée par une ou plusieurs relations (table) et chacune est constituée enregistrements (lignes) et d'attributs (colonnes). Chaque attribut a un type de données et l'on peut spécifier des contraintes dessus.

II. Les types de données et contraintes d'intégrité sur MySQL

1. Les types de données

Un **type de données** indique :

- Le type de valeurs qu'un attribut peut contenir.
- Les opérations qui peuvent être effectuées sur cet attribut.

Les types de données couramment utilisés dans MySQL sont les types **numériques**, les types de **date et d'heure**, et les types de **chaîne de caractères**.

II. Les types de données et contraintes d'intégrité sur MySQL

1. Les types de données

Type de données	Description
Chaînes de caractères	
CHAR(n)	Spécifie des données de type caractère de longueur n. CHAR est de longueur fixe, ce qui signifie que déclarer CHAR (5) implique de réserver des espaces pour 5 caractères.
VARCHAR(n)	Spécifie des données de type caractère de longueur **maximale** n. Contrairement à CHAR, VARCHAR est un type de données de longueur variable. En d'autres termes, déclarer VARCHAR (30) signifie que la longueur maximale de la donnée est de 30 caractères. L'on utilisera **TEXT** pour des chaînes plus longues.

II. Les types de données et contraintes d'intégrité sur MySQL

1. Les types de données

Type de données	Description
Types numériques	
INT	Spécifie une valeur entière. Chaque valeur occupe 4 octets de stockage. Pour de petites valeurs (2 octets), on peut utiliser **SMALLINT** et pour de plus grandes valeurs (8 bits), on utilisera **BIGINT**.
DECIMAL(n, d)	Stocke des valeurs décimales de longueur totale n et une partie décimale d. **NUMERIC(n, d)** peut être utilisé pareillement.
FLOAT	Utilisé pour stocker des nombres à virgule flottante avec précision simple. On utilisera **DOUBLE** ou **REAL** pour des nombres avec précision double.

II. Les types de données et contraintes d'intégrité sur MySQL

1. Les types de données

Type de données	Description
Types dates	
DATE	Utilisé pour stocker des dates au format « AAAA-MM-JJ ». On pourra utiliser **YEAR** pour stocker uniquement les années.
TIME	Stocke le temps sur le format « HH:MM:SS ».
DATETIME	Utilisé pour stocker des dates et des heures sur le format « AAAA-MM-JJ HH-MM-SS ». On utilisera **TIMESTAMP** pour spécifier le fuseau horaire (celui du SGBD).

II. Les types de données et contraintes d'intégrité sur MySQL

1. Les types de données

Type de données	Description
Types booléens	
TINYINT(1)	C'est le type par défaut utilisé pour représenter les données de type booléens. Il existe des synonymes : **BOOL** et **BOOLEAN.** **La valeur 0 est considérée comme « FALSE » (faux) tandis que toute valeur non nulle sera considérée comme « TRUE » (vrai).**

II. Les types de données et contraintes d'intégrité sur MySQL

2. Les contraintes d'intégrité

Une **contrainte d'intégrité** est une restriction ou condition sur les types de données qu'un attribut peut avoir. Elle est utilisée pour garantir l'exactitude et la fiabilité des données. Il n'est cependant pas obligatoire de définir des contraintes pour tous les attributs d'une table.

II. Les types de données et contraintes d'intégrité sur MySQL

2. Les contraintes d'intégrité

Contrainte	Description
NOT NULL	Garantit qu'une colonne ne peut pas avoir de valeur **NULL** (valeur manquante différente de 0 ou espace).
UNIQUE	Assure que toutes les valeurs d'une colonne sont distinctes.
DEFAULT	Définit une valeur par défaut pour une colonne si aucune valeur n'est fournie.
PRIMARY KEY	Indique une colonne ou ensemble de colonnes permettant d'identifier de manière unique chaque enregistrement d'une table.
FOREIGN KEY	Indique une colonne qui fait référence à la valeur d'un attribut défini comme clé primaire dans une autre table.
CHECK(condition)	Assure que toutes les valeurs d'une condition satisfont à une condition spécifiée.

III. Le langage SQL

SQL (**Structured Query Language**) est un langage de requête déclaratif standardisé utilisé pour gérer et manipuler des bases de données relationnelles. Il se compose de plusieurs sous-ensembles qui permettent d'exécuter différentes opérations sur les données : **DDL, DML, DQL, DCL, TCL**.

Dans cette première partie sur SQL, nous nous intéresserons aux trois premiers sous-ensembles. Les autres auront des parties dédiées.

III. Le langage SQL

1. Data Definition Language (DDL)

LDD en français pour **Langage de Définition de Données** est utilisé pour définir les éléments du schéma relationnel de la base de données. Plus simplement, il permet de définir et modifier la structure des objets de la base de données tels que la base de données elle-même, les relations (tables), les schémas, ...

III. Le langage SQL

1. Data Definition Language (DDL)

a. Création

- **Base de données** : `CREATE DATABASE nom_de_la_base_de_données;`

Exemple : `CREATE DATABASE pharmacie;`

Table :

```
CREATE TABLE nom_de_la_table(
        nom_colonne_1 type_colonne_1 contrainte(s)_colonne_1,
        nom_colonne_2 type_colonne_2 contrainte(s)_colonne_2,
                        …
        nom_colonne_n type_colonne_n contrainte(s)_colonne_n,autres_contraintes
)engine="nom_du_moteur";
```

NB : Un **moteur de stockage de base de données** est responsable de la façon dont les données sont stockées, récupérées et gérées dans la base de données. Nous utiliserons **InnoDB** dans ce cours car il prend en compte les transactions. Les autres moteurs communément utilisés sont : **MyIsam, CSV, Memory**, …

III. Le langage SQL

1. Data Definition Language (DDL)

- **Table** :

Exemple : Considérons la table « clients »

Nom du Champ	Description	Type	Contrainte(s)
ID_client	Identifiant unique du client	INT	PRIMARY KEY, AUTO_INCREMENT
nom_client	Nom du client	VARCHAR(100)	NOT NULL
prenom_client	Prénom du client	VARCHAR(100)	NOT NULL
adresse	Adresse du client	VARCHAR(255)	
telephone	Numéro de téléphone	VARCHAR(15)	
date_naissance	Date de naissance	DATE	

NB : **AUTO_INCREMENT** est utilisée pour générer automatiquement des valeurs uniques pour une colonne, généralement une colonne de clé primaire, à chaque insertion d'une nouvelle ligne dans une table .

III. Le langage SQL

1. Data Definition Language (DDL)

- **Table** :

Exemple : En prenant en compte la description précédente, la commande pour créer la table sera la suivante:

```
CREATE TABLE clients (
    ID_client INT AUTO_INCREMENT PRIMARY KEY,
    nom_client VARCHAR(100) NOT NULL,
    prenom_client VARCHAR(100) NOT NULL,
    adresse VARCHAR(255),
    telephone VARCHAR(15),
    date_naissance DATE
);
```

Exercice : Créer les autres tables de la base de données en se basant sur le fichier de description des tables.

III. Le langage SQL

1. Data Definition Language (DDL)

b. Modification des éléments d'une table

La mise à jour des éléments d'une table est introduite par la commande :

```
ALTER TABLE nom_de_la_table;
```

- **Ajouter un attribut**

```
ALTER TABLE ancient_nom_table RENAME TO nouveau_nom_table;
```

- **Ajouter un attribut**

```
ALTER TABLE nom_table ADD nom_attribut TYPE;
```

- **Ajouter une clé primaire**

```
ALTER TABLE nom_table ADD PRIMARY KEY (nom_clé);
```

- **Ajouter une clé étrangère**

```
ALTER TABLE nom_table ADD FOREIGN KEY(nom_attribut) REFERENCES table_de_reference (nom_attribut);
```

III. Le langage SQL

1. Data Definition Language (DDL)

b. Modification des éléments d'une table

- **Ajouter une contrainte UNIQUE sur un attribut**

```
ALTER TABLE nom_table ADD UNIQUE (nom_attribut);
```

- **Modifier le type d'un attribut**

```
ALTER TABLE table_name MODIFY nom_attribut TYPE;
```

- **Modifier une contrainte sur un attribut**

```
ALTER TABLE nom_table MODIFY nom_attribut TYPE contrainte;
```

- **Ajouter une valeur par défaut à un attribut**

```
ALTER TABLE nom_table MODIFY nom_attribut TYPE DEFAULT valeur_défault;
```

III. Le langage SQL

1. Data Definition Language (DDL)

c. Suppression

- **Supprimer une base de données**

```
DROP DATABASE nom_base_de_donnees;
```

- **Vider une table**

```
TRUNCATE TABLE nom_de_la_table;
```

- **Suppression une table**

```
DROP TABLE nom_de_la_table;
```

- **Supprimer un attribut**

```
ALTER TABLE nom_table DROP nom_attribut;
```

- **Supprimer une clé primaire**

```
ALTER TABLE nom_table DROP PRIMARY KEY;
```

- **Supprimer une clé étrangère**

```
ALTER TABLE nom_de_la_table DROP FOREIGN KEY nom__contrainte;
ALTER TABLE nom_table DROP nom_attribut;
```

III. Le langage SQL

2. Data Manipulation Language (DML)

La création d'une table concerne sa structure seule mais elle ne contient aucune donnée. Le **LMD** (**Langage de Manipulation de Données**) est utilisé pour remplir et manipuler les tables de la base de données.

La manipulation des données signifie soit **l'insertion** de nouvelles données, la **modification** et la **suppression** de données existantes.

III. Le langage SQL

2. Data Manipulation Language (DML)

a. Insertion de nouvelles données

Syntaxe de base :

```
INSERT INTO nom_table (colonne_1, colonne_2, ... , colonne_n)
VALUES(valeur_1, valeur_2, ... , valeur_n);
```

pour une unique ligne.

```
INSERT INTO nom_table (colonne_1, colonne_2, ... , colonne_n)
VALUES   (valeur_1, valeur_2, ... , valeur_n),
         (valeur_1_1, valeur_1_2, ... , valeur_m),
                    …
         (valeur_z_1, valeur_z_2, ... , valeur_z);
```

pour plusieurs lignes.

III. Le langage SQL

2. Data Manipulation Language (DML)

a. Insertion de nouvelles données

Exemple :

```
INSERT INTO clients (nom_client, prenom_client, adresse, telephone, date_naissance)
VALUES
('Nguemeta', 'Paul', 'Rue de la Santé, Yaoundé', '679000001', '1985-05-12'),
('Tchako', 'Marie', 'Avenue de la Paix, Douala', '679000002', '1990-11-03'),
('Ngongang', 'Jean', 'Quartier Bafoussam', '679000003', '1980-08-19'),
('Tita', 'Sylvie', 'Rue de l\'Hôpital, Garoua', '679000004', '1995-02-27'),
('Mvondo', 'Pierre', 'Boulevard de la Réunification, Yaoundé', '679000005',
'1988-03-15');
```

Exercice : Effectuer les insertions pour les autres tables puis exporter la base de données.

III. Le langage SQL

2. Data Manipulation Language (DML)

b. Mise à jour de données

Syntaxe :

```
UPDATE nom_table
SET attribut_1 = valeur_1, attribut_2 = valeur_2, ...
WHERE condition;
```

Exemple : Modifier les noms de tous les clients ayant pour prénom « Jean » par « Kamga ».

```
UPDATE clients
SET nom_client = "Kamga"
WHERE prenom_client = "Jean";
```

Exercice : Modifier les noms commerciaux des medicaments de catégorie 1 par "Efferalgan".

III. Le langage SQL

2. Data Manipulation Language (DML)

c. Suppression de données

Syntaxe :

```
DELETE nom_table
WHERE condition;
```

Exemple : Supprimer les clients ayant pour nom « Kamga ».

```
DELETE clients
WHERE nom_client = “Kamga”;
```

Exercice : Supprimer tous les medicaments ayant pour nom commercial “Efferalgan”.

NB : TOUJOURS PRÉCISER LA CONDITION LORS D’UNE OPÉRATION DE MISE À JOUR ET SUPPRESSION AFIN D’ÉVITER UNE OPÉRATION DE MASSE.

III. Le langage SQL

3. Data Query Language (DQL)

Le **LID** (**Langage d'Interrogation de Données**) est utilisé pour interroger et récupérer les données de la base de données. Il est parfois inclus dans le LMD.

III. Le langage SQL

3. Data Query Language (DQL)

a. Requêtes basiques

La commande de base du LID est introduite par la commande **SELECT**.

- **Sélection de colonnes**

```
SELECT colonne_1, ... , colonne_2
FROM nom_table;
```

NB : L'on peut utiliser * lorsqu'on souhaite récupérer toutes colonnes d'une relation.

Exercice : Trouver les noms des clients.

- **Filtre (Restriction)**

```
SELECT colonne_1, ... , colonne_2
FROM nom_table
WHERE condition;
```

Exercice : Trouver les informations des commandes dont le statut est « Livrée ».

III. Le langage SQL

3. Data Query Language (DQL)

a. Requêtes basiques (Opérateurs utilisables avec la clause WHERE)

- **Opérateurs de comparaison (<, >, <> ou !=, <=, >=)** : Utilisés pour comparer les valeurs d'une colonne avec valeur spécifique ou une autre colonne.
- **Opérateurs logiques (AND, OR, NOT)** : Utilisés pour combiner plusieurs conditions.
- **Opérateurs spéciaux (BETWEEN, IN, LIKE, IS NULL, IS NOT NULL)** : Ils offrent des moyens avancés pour filtrer les données en fonction de plages de valeurs de modèles spécifiques ou de la présence de valeurs **NULL**.

III. Le langage SQL

3. Data Query Language (DQL)

a. Requêtes basiques (Opérateurs utilisables avec la clause WHERE)

Exercice

1. Rechercher tous les médicaments qui appartiennent à la catégorie "Antibiotique" (ID_categorie = 2).
2. Rechercher tous les clients nés après le 1er janvier 1990.
3. Rechercher toutes les commandes effectuées avant le 1er décembre 2023.
4. Rechercher tous les médicaments de la catégorie 1 dont le prix est inférieur à 500 FCFA.
5. Rechercher tous les clients vivant soit à Douala, soit à Yaoundé.
6. Rechercher tous les clients dont l'adresse n'est pas renseignée.
7. Rechercher tous les médicaments dont le prix est compris entre 500 et 2000 FCFA.
8. Trouver les médicaments ayant pour nom commercial « Doliprane », « Aspirine » ou « Augmentin ».
9. Lister les adresses et les noms des fournisseurs dont le numéro de téléphone se termine par « 4 ».

III. Le langage SQL

3. Data Query Language (DQL)

a. Requêtes basiques (Complément)

- **Limiter le nombre de résultats : LIMIT**

```
SELECT colonne_1, ... , colonne_2
FROM nom_table
. . .
LIMIT nombre_de_resultats_souhaite;
```

- **Grouper les données suivant une ou plusieurs colonne : GROUP BY**

```
SELECT colonne_1, ... , colonne_2
FROM nom_table
GROUP BY nom_de_colonne_de_groupement;
```

- **Filtrer les groupes : HAVING**

```
SELECT colonne_1, ... , colonne_2
FROM nom_table
GROUP BY nom_de_colonne_de_groupement
HAVING condition;
```

III. Le langage SQL

3. Data Query Language (DQL)

b. Jointure interne

Syntaxe générale :

```
SELECT colonne_1, ... , colonne_2
FROM table_1
INNER JOIN table_2 ON table_1.colonne_table_1 = table_2.colonne_table_2;
```

Exemple : Trouver les noms des clients ayant passer des commandes.

```
SELECT nom_client
FROM clients
INNER JOIN commandes ON clients.ID_client = commandes.ID_client;
```

Exercice : Rechercher les noms des médicaments commandés ainsi que ceux de leurs fournisseurs.

III. Le langage SQL

3. Data Query Language (DQL)

b. Autres jointures

Bien que la jointure interne soit la plus utilisée, il en existe plusieurs autres :

- **Jointures externes** : **LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN** (MySQL ne supportant pas directement FULL OUTER JOIN, il est possible d'obtenir le résultat escompté en combinant LEFT JOIN et RIGHT JOIN à partir de la commande **UNION**).
- **Jointure croisée** : **CROSS JOIN** (correspond à l'opération de produit cartésien).
- **Jointure auto** : **SELF JOIN.**

III. Le langage SQL

3. Data Query Language (DQL)

c. Fonctions d'agrégation

Les **fonctions d'agrégation** en SQL sont utilisées pour effectuer des calculs sur un ensemble de valeurs et retourner une seule valeur.

Elles sont couramment utilisées avec la clause GROUP BY pour regrouper les données en fonction d'une ou plusieurs colonnes. Les plus utilisées sont :

- **COUNT()** : Compter des lignes.
- **SUM()** : Calculer la somme des valeurs d'une colonne numérique.
- **AVG()** : Calculer la moyenne des valeurs d'une colonne numérique.
- **MIN() / MAX()** : Trouver la valeur minimale/maximale d'une colonne.

Devoir (version française)

1. Déterminer le nombre de clients enregistrés dans la base de données.
2. Calculer le montant des ventes pour chaque commande.
3. Déterminer le prix moyen des médicaments dans chaque catégorie.
4. Trouver le prix le plus bas et le plus élevé parmi tous les médicaments.
5. Compter le nombre de médicaments fournit par chaque fournisseur.
6. Calculer le montant total des commandes de chaque client.
7. Compter le nombre de commandes par statut.
8. Obtenir les noms commerciaux des médicaments commandés ainsi que les noms de leurs fournisseurs. Assurez-vous de supprimer les doublons.
9. Déterminer combien de médicaments sont expirés dans chaque catégorie.
10. Rechercher tous les clients et les informations sur leurs commandes, y compris ceux qui n'ont pas passé de commande.

Devoir (version française)

L'on a vu dans la partie précédente que $\boldsymbol{R \bowtie_P S = \sigma_p(R \times S)}$ (« Une jointure interne est équivalente à une sélection appliquée à un produit cartésien »), R et S étant des relations et p un prédicat (condition).

Montrer cette égalité en écrivant une requête qui affiche les noms des clients et les identifiants de leurs commandes ainsi que leurs statuts.

Assignment (English version)

1. Determine the number of customers registered in the database.
2. Calculate the sales amount for each order.
3. Determine the average price of medications in each category.
4. Find the lowest and highest price among all medications.
5. Count the number of medications supplied by each supplier.
6. Calculate the total amount of orders for each customer.
7. Count the number of orders by status.
8. Get the commercial names of ordered medications and the names of their suppliers. Make sure to remove duplicates.
9. Determine how many medications are expired in each category.
10. Fetch all clients and their order information, including those who have not placed any orders..

Assignment (English version)

We saw in the previous section that $\boldsymbol{R \bowtie_P S = \sigma_p}$ ('An inner join is equivalent to a selection applied to a Cartesian product'), where R and S are relations and p is a predicate (condition).

Show this equality by writing a query that displays the names of clients and the identifiers of their orders as well as their statuses."