PARTIE 1 : Modèles de données relationnels

Dans une base de données relationnelle, les tables ont des lignes uniques. Cette unicité permet d'établir des relations entre les tables, généralement par le biais de **clés primaires** et de **clés étrangères**.

Clé primaire : une clé primaire est la donnée qui permet d'identifier de manière unique un enregistrement dans une table. La clé primaire d'une table peut être constituée d'une ou plusieurs colonnes

Clé étrangère : une colonne dans une table, qui fait référence à la clé primaire dans une autre table.

Un schéma de base de données est une documentation de sa structure.
Il comprend :

- Les noms des **tables**
- Les noms des **colonnes** (ou champs) de chaque table
- Le **type** de données de chaque champ (float, VarChar, integer etc.)
- La définition des **contraintes**, des valeurs par défaut, des actions d'intégrité référentielle...
- Les **relations** entre les tables, à travers la spécification des clés primaires, des clés étrangères.

Il existe trois types de relations entre deux tables :

- **One-to-one** : typiquement équivalent à la division d'une seule table et à la conservation de la clé primaire.
- **One-to-many** : ex. commandes et utilisateurs.
- **Many-to-many** : ex. commandes et produits.

Dans une base de données relationnelle, une relation many-to-many entre deux tables nécessitera l'existence d'une troisième table intermédiaire pour faire le pont. Cette table de jonction aura une relation one-to-many avec chacune des tables d'origine.

PARTIE 2 : Les opérations basiques en SQL

Les opérateurs de base permettent d'extraire des données d'un tableau, avec la possibilité de filtrer les colonnes et les lignes.

Clause SELECT

- Introduction obligatoire à toute requête d'extraction de données

- Permet la sélection / de filtrer des colonnes

- * est équivalent à la liste de toutes les colonnes

- Est suivie de toutes les colonnes que vous souhaitez afficher
    - Un scalaire ou un texte est considéré comme une colonne (le même scalaire/texte étant répété sur toutes les lignes).

- Les colonnes doivent être séparées par une virgule

- Syntaxe : SELECT colonne_1, colonne_2, colonne_3, ..., colonne_n

Clause FROM

- Indique dans quelle table les colonnes mentionnées doivent être sélectionnées

- Directement après le nom de la table, un alias peut lui être attribué.

- Syntaxe : FROM nom_table AS nom_table_alias

Clause WHERE

- Indique les conditions sur lesquelles les lignes doivent être sélectionnées
    - Dans cette clause, les conditions doivent porter sur les données telles qu'elles sont dans la table (pas d'agrégation).

- Une condition peut concerner autant de colonnes que nécessaire

- Il peut y avoir autant de conditions que nécessaire (séparées par AND ou OR).

- Syntaxe : WHERE condition_1 AND/OR condition_2 AND/OR condition_3

- Les conditions peuvent être : =, ≠, >,<, IN, EXISTS, LIKE %, BETWEEN

Fonctions d'agrégation

- Count()
- Sum()
- AVG()
- MIN()
- MAX()

Peuvent être combinées avec une clause `GROUP BY` pour agréger les résultats en fonction de la valeur de certains champs.

Clause GROUP BY

L'instruction `GROUP BY` permet de regrouper les lignes par une valeur qu'elles ont en commun.

Elle est souvent utilisée avec les fonctions d'agrégation (`COUNT()`, `MAX()`, `MIN()`, `SUM()`, `AVG()`) pour regrouper le jeu de résultats par une ou plusieurs colonnes.

Par exemple : "trouver le nombre de clients dans chaque pays". On ferait pour cela un COUNT(id_client) et un GROUP BY country

Clause HAVING

Une clause HAVING en SQL spécifie qu'une instruction SQL SELECT ne doit renvoyer que les lignes dont les valeurs groupées répondent aux conditions spécifiées.

Après l'opération d'agrégation, la clause HAVING est appliquée, filtrant les lignes qui ne correspondent pas aux conditions spécifiées.

La clause WHERE pose des conditions sur les colonnes sélectionnées, tandis que la clause HAVING pose des conditions sur les groupes créés par la clause GROUP BY.

Alias AS

Les alias SQL sont utilisés pour donner à une table, ou à une colonne dans une table, un nom temporaire.

Les alias sont souvent utilisés pour rendre les noms de colonnes plus lisibles.

Un alias n'existe que pour la durée de cette requête.

Un alias est créé avec le mot-clé `AS` (facultatif pour les tables dans les clauses FROM/JOIN).

Clause ORDER BY

Le mot-clé `ORDER BY` est utilisé pour trier le jeu de résultats dans un ordre croissant ou décroissant.

Par défaut, le mot-clé `ORDER BY` trie les enregistrements dans l'ordre croissant `ASC`. Pour trier les enregistrements par ordre décroissant, utilisez le mot-clé `DESC`.

Clause LIMIT

- Utilisée pour tester vos requêtes, pour avoir un aperçu d'une table sans interroger tout son contenu.
- Utilisée pour obtenir les meilleurs résultats
- Toujours placée **en dernier** dans votre requête