Fiche mémo SQL

Présentation

- SQL = Structured Query Language
- exploiter des bases de données relationnelles. L'information dans une base de données relationnelle est organisée en tableaux à deux dimensions appelés des relations ou des tables
- Un SGBDR ou RDBMS (Relational DataBase Managment System) permet de manipuler, créer et modifier l'organisation des données selon un langage normalisé ; le SQL

1. Langage de manipulation des données

- **SELECT**: Récupérer des données depuis une table.
  Ex: SELECT * FROM clients WHERE id > 3;
- **DISTINCT**: Éliminer les doublons dans les résultats.
  Ex: SELECT DISTINCT nom FROM clients;
- **WHERE**: Filtrer les résultats selon une condition.
  Ex: SELECT * FROM clients WHERE tel IS NOT NULL;
- **ORDER BY**: Trier les résultats par ordre croissant (ASC) ou décroissant (DESC).
  Ex: SELECT * FROM clients ORDER BY nom ASC;
- **GROUP BY** et **HAVING**: Regrouper les résultats et appliquer des conditions sur les groupes.
  Ex: SELECT COUNT(*), ville FROM clients GROUP BY ville HAVING COUNT(*) > 1;

2. Opérations sur les tables

- **CREATE TABLE**: Créer une table.
  Ex: CREATE TABLE clients (id INT PRIMARY KEY, nom VARCHAR(255));
- **ALTER TABLE**: Modifier la structure d'une table (ajouter ou supprimer une colonne).
  Ex: ALTER TABLE clients ADD dateNaissance DATE;
- **DROP TABLE**: Supprimer une table et son contenu.
  Ex: DROP TABLE clients;
- **TRUNCATE TABLE**: Supprimer toutes les lignes d'une table sans supprimer sa structure.
  Ex: TRUNCATE TABLE clients;

3. Gestion des jointures

- **INNER JOIN**: Récupérer les enregistrements ayant une correspondance dans les deux tables.
  Ex: SELECT * FROM clients INNER JOIN adresses ON clients.id = adresses.idClient;
- **LEFT JOIN**: Récupérer tous les enregistrements de la table de gauche même s'il n'y a pas de correspondance dans celle de droite.
  Ex: SELECT * FROM clients LEFT JOIN adresses ON clients.id = adresses.idClient;
- **RIGHT JOIN**: Récupérer tous les enregistrements de la table de droite même s'il n'y a pas de correspondance dans celle de gauche.
  Ex: SELECT * FROM clients RIGHT JOIN adresses ON clients.id = adresses.idClient;

4. Fonctions d’agrégation

- **MIN()**: Obtenir la valeur minimale.
  Ex: SELECT MIN(codePostal) FROM adresses;
- **MAX()**: Obtenir la valeur maximale.
  Ex: SELECT MAX(codePostal) FROM adresses;
- **COUNT()**: Compter le nombre d'enregistrements.
  Ex: SELECT COUNT(*) FROM clients;
- **SUM()**: Calculer la somme des valeurs.
  Ex: SELECT SUM(codePostal) FROM adresses;
- **AVG()**: Calculer la moyenne des valeurs.
  Ex: SELECT AVG(codePostal) FROM adresses;

5. Opérateur LIKE et WildCards

- **LIKE**: Rechercher des motifs spécifiques dans les chaînes de caractères avec % pour 0 ou plusieurs caractères, et _ pour un caractère.
  Ex: SELECT * FROM clients WHERE nom LIKE '%toto';

6. Contrôle d'existence

- **IF OBJECT_ID('nomObjet') IS NULL CREATE**: Tester l'existence d'un objet avant de le créer.
  Ex: IF OBJECT_ID('clients') IS NULL CREATE TABLE clients (id INT PRIMARY KEY, nom VARCHAR(255));
- **CREATE ... IF NOT EXISTS**: Créer un objet uniquement s'il n'existe pas déjà.
  Ex: CREATE TABLE IF NOT EXISTS clients (id INT PRIMARY KEY, nom VARCHAR(255));
- **DROP TABLE IF EXISTS**: Supprimer une table si elle existe.
  Ex: DROP TABLE IF EXISTS clients;

Jointure croisée : Permet de combiner deux tables sans condition explicite, générant le produit cartésien des deux tables (exemple : 22 clients et 40 adresses donnent 880 lignes).

- **Jointure interne (INNER JOIN)** : Utilisée pour sélectionner uniquement les lignes ayant une correspondance dans les deux tables. Par exemple, lier les clients et les adresses par leur id.
- **Jointure externe (LEFT, RIGHT, FULL JOIN)** :
  - **LEFT JOIN** : Toutes les lignes de la table de gauche (clients) seront dans le résultat, même si elles n'ont pas de correspondance dans la table de droite (adresses).
  - **RIGHT JOIN** : L'inverse de LEFT JOIN, toutes les lignes de la table de droite seront présentes, avec des valeurs NULL si pas de correspondance.
  - **FULL JOIN** : Combine LEFT et RIGHT JOIN, mais MySQL ne supporte pas FULL JOIN directement.

2. Fonctions d'agrégation

- **MIN() et MAX()** : Renvoient la valeur minimale ou maximale d’une colonne.
- **COUNT()** : Compte le nombre de lignes ou de valeurs non NULL.
- **SUM()** : Renvoie la somme des valeurs numériques d'une colonne.
- **AVG()** : Calcule la moyenne des valeurs d'une colonne numérique.