OBJECTIFS

Objectif Global

Se familiariser avec les différents types de requêtes SQL

Objectifs pédagogiques

- Savoir utiliser les requêtes de base SQL
- Savoir mettre en oeuvre du scripting basique SQL

Démarche pédagogique

Vous allez découvrir et mettre en œuvre des requêtes sql pour gérer un schéma de données.
A l'issue de cette partie vous serez en mesure d'utiliser une base de données et de récupérer les informations qu'elle contient et les mettre en forme.

Compétences

- Créer un mémo
- Écrire les requêtes de bases SQL.

MODALITÉS

Durée

1,0J

ITÉRATION 1

Requêtes

Modalités

- Travail individuel en autonomie

Livrables

Les scripts SQL contenant les exemples de requêtes courantes

Objectifs

Découvrir les instructions SQL de base

Rappel de la structure de la base de données

Vous utiliserez la base de données de Numcity que vous avez créée précédemment. Chaque étudiant devra adapter ses requêtes en fonction du schéma spécifique qu'il a conçu. Cela implique de bien comprendre les relations et contraintes définies dans vos tables, ce qui est essentiel pour écrire des requêtes efficaces et précises.

Préliminaires

Avant de commencer, assurez-vous d'avoir accès à PostgreSQL et d'avoir créé la base de données de Numcity. Si nécessaire, référez-vous au site officiel de PostgreSQL pour des ressources sur l'installation et la configuration. Ce cours suppose une installation fonctionnelle.

Partie 1 : Requêtes d'insertion (INSERT)

Explications

Les requêtes INSERT servent à ajouter de nouvelles données dans une table. Elles permettent de spécifier explicitement les colonnes et les valeurs associées. Cette opération est essentielle pour enrichir votre base de données.

Syntaxe générale :

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

Options avancées :

- Insertion multiple : ajoutez plusieurs lignes en une seule requête.
- Valeurs par défaut : utilisez DEFAULT si certaines colonnes ont une valeur par défaut définie dans la table.

Exemple

Ajoutons une nouvelle station :

```
INSERT INTO Stations (nom, adresse, latitude, longitude, type_station, nombre_places)
 VALUES ('Station Centre', '10 rue des Lilas', 48.8566, 2.3522, 'recharge', 20);
```

Ajoutons plusieurs stations en une seule requête :

```
INSERT INTO Stations (nom, adresse, latitude, longitude, type_station, nombre_places)
 VALUES
('Station Nord', '1 avenue du Parc', 48.867, 2.354, 'parking', 15),
('Station Sud', '5 boulevard Saint-Michel', 48.841, 2.337, 'recharge', 10);
```

Exercice

Commencez par identifier des exemples de données complètes et incomplètes pour chaque table dans votre base. Insérez des enregistrements en vous assurant d'utiliser différentes combinaisons de colonnes remplies ou laissées vides, lorsque cela est permis. Assurez-vous de tester des cas avec des valeurs par défaut et des colonnes facultatives. Par exemple, insérez des informations pour une table représentant des utilisateurs en laissant le champ email vide, ou en spécifiant seulement leur prénom et nom. Répétez cette procédure pour chaque table et notez vos observations dans un mémo détaillé.

Réalisez également toutes les requêtes qui vous semblent utiles pour faciliter l'exploitation de la base de données à l'avenir.

Partie 2 : Requêtes de mise à jour (UPDATE)

Explications

Les requêtes UPDATE permettent de modifier des enregistrements existants. Elles sont particulièrement utiles pour corriger ou actualiser des données. La syntaxe nécessite une clause WHERE pour éviter de mettre à jour toutes les lignes de la table accidentellement.

Syntaxe générale :

```
UPDATE nom_table SET colonne1 = valeur1, colonne2 = valeur2 WHERE condition;
```

Options avancées :

- Mise à jour conditionnelle sur plusieurs colonnes.
- Utilisation de sous-requêtes pour attribuer des valeurs basées sur d'autres tables.

Exemple

Modifions le nombre de places disponibles dans une station :

```
UPDATE Stations SET nombre_places = 25 WHERE nom = 'Station Centre';
```

Changeons le type de station pour un cas particulier :

UPDATE Stations SET type_station = 'parking' WHERE id_station = 2;

Exercice

Identifiez un emplacement nécessitant une mise à jour et modifiez ses coordonnées pour refléter un changement récent. Changez le type d'un emplacement en une autre catégorie et expliquez dans votre mémo pourquoi ce changement est pertinent, en prenant exemple sur des modifications réalistes. Effectuez des mises à jour conditionnelles impliquant plusieurs champs à la fois et testez les résultats pour vérifier leur précision.

Réalisez également toutes les requêtes qui vous semblent utiles pour faciliter l'exploitation de la base de données à l'avenir.

Partie 3 : Requêtes de suppression (DELETE)

Explications

Les requêtes DELETE permettent de supprimer des lignes spécifiques dans une table. Ces requêtes doivent être utilisées avec précaution, car elles sont définitives.

Syntaxe générale :

DELETE FROM nom_table WHERE condition;

Options avancées :

- Suppression conditionnelle avancée avec des sous-requêtes.
- Utilisation conjointe avec RETURNING pour récupérer les données supprimées.

Exemple

Supprimons une réservation obsolète :

DELETE FROM Réservations WHERE id_réservation = 1;

Exercice

Rédigez une requête pour supprimer un enregistrement ne répondant plus aux critères définis, par exemple un emplacement qui ne comporte plus aucune disponibilité. Vérifiez que la condition

est bien respectée avant d'exécuter la suppression. Ensuite, utilisez une clause RETURNING pour afficher les détails des données qui ont été supprimées, afin de valider les modifications effectuées. Notez vos observations dans votre mémo et expliquez les étapes suivies.

Réalisez également toutes les requêtes qui vous semblent utiles pour faciliter l'exploitation de la base de données à l'avenir.

Partie 4 : Requêtes de sélection (SELECT)

Explications

Les requêtes SELECT sont le pilier de l'interrogation des bases de données. Elles permettent de récupérer des données spécifiques ou de faire des analyses complexes.

Syntaxe générale :

SELECT colonne1, colonne2 FROM nom_table WHERE condition;

Options avancées :

- Alias de colonnes pour une meilleure lisibilité.
- Utilisation des fonctions comme COUNT(), AVG(), MAX(), etc.
- Sous-requêtes dans le SELECT ou le WHERE.

Exemple simple

Affichons toutes les stations :

SELECT * FROM Stations;

Exemple avancé

Affichons les stations de recharge avec plus de 10 places :

SELECT nom, adresse FROM Stations WHERE type_station = 'recharge' AND nombre_places > 10;

Exercice

Identifiez et énumérez toutes les réservations associées à un utilisateur spécifique en utilisant une requête adaptée. Rédigez une autre requête permettant d'afficher uniquement les emplacements

disposant de peu de disponibilités, par exemple moins de cinq unités libres. Enfin, combinez plusieurs critères dans une requête afin d'obtenir des résultats plus précis et significatifs pour répondre à des besoins spécifiques.

Réalisez également toutes les requêtes qui vous semblent utiles pour faciliter l'exploitation de la base de données à l'avenir.

Partie 5 : Requêtes complexes

Les jointures (JOIN)

Les jointures permettent de relier des tables entre elles pour extraire des informations combinées. Voici un exemple classique de jointure :

```
SELECT Utilisateurs.nom, Utilisateurs.prenom, Stations.nom AS station
 FROM Réservations
 JOIN Utilisateurs ON Réservations.id_utilisateur = Utilisateurs.id_utilisateur
 JOIN Stations ON Réservations.id_station = Stations.id_station;
```

Les agrégations

Les fonctions d'agrégation permettent de calculer des statistiques sur les données. Elles s'utilisent souvent avec GROUP BY.

Exemple d'agrégation

```
SELECT type_station, COUNT(*) AS nombre_stations
 FROM Stations
 GROUP BY type_station;
```

Exercices

Rédigez une requête permettant de calculer le nombre total de réservations effectuées par chaque utilisateur. Rédigez une autre requête visant à afficher uniquement les utilisateurs associés à un enregistrement spécifique dans la base de données. Enfin, combinez plusieurs tables et appliquez des fonctions d'agrégation pour générer un rapport synthétique regroupant des statistiques clés sur les données.

Réalisez également toutes les requêtes qui vous semblent utiles pour faciliter l'exploitation de la base de données à l'avenir.

Ressources supplémentaires

- Documentation officielle PostgreSQL
- SQL Tutorial
- Exercices SQL interactifs
- Cheat sheet SQL
- Advanced SQL Queries Tutorials by Mode Analytics
- LeetCode Database Questions
- SQL Practice Problems
- Aller plus loin : https://sqlzoo.net/wiki/SQL_Tutorial

N'oubliez pas de remplir votre mémo avec des exemples personnels tout au long du cours. Bonne pratique et exploration !