Contrôle Continu — PostgreSQL

Nom : Nadège Kangni-soukpe

Formations: M1 IA

Exercice 1 – Questions théoriques

1. Quelle est la différence entre un rôle et un utilisateur dans PostgreSQL ?

Dans PostgreSQL, un rôle est une entité qui représente un utilisateur ou un groupe d'utilisateurs. Y

La principale différence

U**n rôle peut avoir plusieurs utilisateurs** associés, et il peut être attribué des permissions spécifiques sur des objets de la base de données (comme les tables ou les vues).

Un **utilisateur**, quant à lui, est un rôle qui se voit attribuer des privilèges pour se connecter à la base de données et l'utiliser. En résumé, un utilisateur est un rôle avec la capacité de se connecter à la base.

2. Expliquez l'utilité du fichier `pg_hba.conf`.

Le fichier `pg_hba.conf` dans PostgreSQL est un fichier de configuration crucial qui contrôle **l'accès aux bases de données** en définissant qui peut se connecter, depuis quelle machine et avec quel type d'authentification.

C'est ce fichier qui protège la base contre les connexions non autorisées.

3. Quelle est la différence entre une sauvegarde logique et une sauvegarde physique ?

Une **sauvegarde logique** consiste à exporter les objets de la base de données (tables, schémas, données) dans un format lisible (souvent un fichier SQL). Elle peut être effectuée avec des outils comme `pg_dump`.

Une **sauvegarde physique** inclut la copie des fichiers système de la base de données (les fichiers de données et de journalisation), ce qui permet une récupération complète du système en cas de crash, et elle se fait généralement via une méthode comme `pg_basebackup.`

4. Donnez deux commandes SQL permettant de sécuriser l'accès à une base PostgreSQL.

- **Commande de création d'un utilisateur avec un mot de passe sécurisé :**

```
CREATE USER admin_etu WITH PASSWORD 'admin123';
```

- **Limiter l'accès par IP (via pg_hba.conf)** :

Pour ce genre de sécurité, on modifie les règles dans pg_hba.conf pour spécifier que seules certaines adresses IP peuvent accéder à la base de données.

5. Qu'est-ce qu'un tablespace ? Pourquoi l'utiliser ?

Un **tablespace** dans PostgreSQL est un emplacement de stockage qui permet de gérer l'espace disque de manière plus flexible.

Pourquoi l'utiliser ?

En créant un tablespace, on peut séparer les données d'une base en différents répertoires physiques pour des raisons de performance ou d'organisation.

6. À quoi sert la commande `EXPLAIN ANALYZE` ?

La commande `EXPLAIN ANALYZE` permet de visualiser le plan d'exécution d'une requête SQL ainsi que le temps qu'elle prend à s'exécuter réellement.

Exemple:

EXPLAIN ANALYZE SELECT * FROM etudiants WHERE nom = 'Dupont';

(Cela permet de voir comment PostgreSQL va exécuter cette requête et où il pourrait y avoir des améliorations à apporter (par exemple en ajoutant un index).

7. Qu'est-ce qu'une transaction ?

Une **transaction** est une séquence d'opérations SQL qui sont traitées comme une seule unité. Si une transaction échoue, aucune des opérations n'est enregistrée dans la base, ce qui garantit l'intégrité des données.

- **BEGIN** : Débute une transaction.
- **COMMIT** : Valide la transaction, enregistrant toutes les modifications effectuées.
- **ROLLBACK** : Annule la transaction, annulant toutes les modifications depuis le `BEGIN`.

8. Quelle est la commande pour créer une base de données et attribuer les droits à un utilisateur ?

La commande pour créer une base de données et donner les droits à un utilisateur est :

```
CREATE DATABASE biblio_db;

CREATE USER biblio_admin WITH PASSWORD 'biblio123';

GRANT ALL PRIVILEGES ON DATABASE biblio_db TO biblio_admin;
```

9. Expliquez les rôles de `pgAudit` et `pg_stat_statements`.

- **pgAudit** : C'est une extension qui permet de suivre les activités dans PostgreSQL. Elle est particulièrement utile pour l'audit et la conformité des données.
- **pg_stat_statements** : C'est une vue qui permet de suivre les performances des requêtes SQL. Elle permet de récupérer des informations sur les requêtes les plus lentes et les plus coûteuses, ce qui est essentiel pour optimiser la base de données.

10. Quelle commande permet d'installer une extension dans PostgreSQL ?

La commande pour installer une extension dans PostgreSQL est :

```
CREATE EXTENSION nom_extension;
```

Par exemple, pour installer `pg_stat_statements`, on utiliserait :

```
CREATE EXTENSION pg_stat_statements;
```