TP : Bases de Données Distribuées avec PostgreSQL et Redis

Objectif :

Mettre en place un système de base de données distribuée utilisant PostgreSQL en mode primaire-secondaire ainsi que Redis pour le stockage en mémoire d'une table spécifique. L'objectif est de comprendre la réplication des bases de données et l'optimisation des performances avec un cache en mémoire avec des BDs hydrides.

Partie 1 : Mise en place de PostgreSQL avec réplication

Dans cette première partie, vous allez installer et configurer PostgreSQL avec une instance primaire et une instance secondaire qui recevra les mises à jour en temps réel.

Étapes :

1. Installer PostgreSQL sur deux machines virtuelles ou conteneurs.
2. Configurer l’instance principale pour activer la réplication.
3. Configurer l’instance secondaire pour recevoir les mises à jour.
4. Vérifier la synchronisation des données entre les instances.

Partie 2 : Utilisation de Redis pour optimiser les performances

Redis sera utilisé comme un cache pour une table spécifique dans PostgreSQL. Cela permettra de réduire la charge sur la base relationnelle en stockant les données en mémoire.

Étapes :

1. Installer Redis sur le serveur ou utiliser l’offre cloud de Redis.
2. Configurer une connexion entre PostgreSQL et Redis.
3. Créer une table « taux de change  » dans PostgreSQL et y insérer des données.
4. Stocker les données de cette table dans Redis.
5. Comparer les temps de requêtage entre PostgreSQL et Redis.

## Partie 3 : Synchronisation des données entre PostgreSQL et Redis

Dans cette partie, nous allons nous assurer que les données stockées dans Redis restent à jour lorsqu'une modification est effectuée dans PostgreSQL.

Quelques pistes possibles :

* Utilisation de déclencheurs (triggers) dans PostgreSQL pour mettre à jour Redis.
* Écoute des changements avec PostgreSQL NOTIFY/LISTEN.
* Comparaison entre les données stockées dans PostgreSQL et celles dans Redis.

## Partie 4 : Expérimentations et Analyse des performances

Expériences à réaliser :

* Développer un WS Rest qui interroge les BDs.
* Effectuer des requêtes répétées sur PostgreSQL et mesurer le temps de réponse.
* Effectuer les mêmes requêtes avec Redis et comparer les performances.
* Simuler une panne de l’instance primaire et observer le comportement du secondaire.