Université Mohamed V de Rabat **Année universitaire 2019-2020**
ENSIAS

TP1 - Bases de données réparties
M. Nassar

Soit la base de données relationnelle ***Comptes*** composée des relations suivantes :

Client (*No*, Nom, Prénom, Adresse, Ville)
Agence (*No*, Nom, Adresse, Ville)
Compte (*No*, Type_Compte_No, DateOuverture, Decouvert_autorise, Solde, *Client_No*, *Agence_No*)
Type_Compte (*No*, Nom, Description)
Operation (*No*, Type_Operation_No, *Compte_No*)
Type_Operation (*No*, Description)

1- Répartition des données : définition des fragments

A partir de la base *Comptes* centralisée déjà mise en œuvre sur la base *ENSIAS1* sur le serveur *Serveur1*, on désire construire une base de données répartie sur les deux sites : *Serveur1 et Serveur2*.

Les règles de répartition ou de fragmentation ont été définies en fonction de certains critères d'utilisation et de manipulation des données.

a- Fragmentation horizontale pour la table **Client** :

- Sur *Serveur1* : la table **Client_1** contenant les clients de Casablanca sans la colonne Ville.
- Sur *Serveur2* : la table **Client_2** contenant les clients de Rabat sans la colonne Ville.

b- Fragmentation horizontale pour la table **Compte** :

- Sur *Serveur1* : la table **Compte_1** avec les comptes appartenant aux clients de Casablanca.
- Sur *Serveur2* : la table **Compte_2** avec les comptes appartenant aux clients de Rabat.

c- Fragmentation horizontale pour la table **Operation** :

- Sur *Serveur1* : la table *Operation_1* correspondant aux opérations des comptes de la table *Compte_1*.
- Sur *Serveur2* : la table *Operation_2* correspondant aux opérations des comptes de la table *Compte_2*.

d- Déplacement complet de la table **Type_Compte** sur *Serveur2* : *Type_Compte_2*

e- Déplacement complet de la table **Type_Operation** sur *Serveur2* : *Type_Operation_2*

f- Les Séquences restent sur *Serveur1*.

2- Création des fragments sur les deux sites

Avec la commande COPY (sqlplus), créer les fragments sur les deux bases : *Serveur1* et *Serveur2*. Vérifier la présence des fragments puis détruire les tables initiales.

COPY [FROM spécification_base1]
[TO spécification_base2]
{APPEND | CREATE | REPLACE | INSERT} nom_table [colonnes]
USING SELECT .....

APPEND : si la table n'existe pas (create + insert) sinon (insert)
CREATE : si la table n'existe pas (create + insert) sinon (erreur)
REPLACE : si la table n'existe pas (create + insert) sinon (DROP + CREATE + INSERT)
INSERT : si la table n'existe pas (erreur) sinon (insert)

Conseil : utiliser l'option REPLACE dans vos COPY.

Vérifier le contenu de tous les fragments en affichant leur contenu.

Effacer les tables initiales de la bd centralisée sur *Serveur1*.

3- Création du lien inter – base (database link)

Sur chaque base (*Serveur1* et *Serveur2*), créer deux **database link** (**dbl_Ensias1** et **dbl_Ensias2**) permettant d'accéder aux objets distants.

Tester les liens établis sur chaque base en accédant aux objets distants dans les deux sens (Ex. : SELECT * FROM client_2@dbl_Ensias2;).

4- Ajout des contraintes de base

Ajouter, sur chaque fragment, les contraintes initiales qui ont disparues :

(1) Les Contraintes de Clé Primaire,
(2) Les Contraintes de Références classiques si la table 'mère' est sur le même site,
(3) Les Contraintes de Références par 'trigger' si la table 'mère' est sur un site distant
Deux trigger :
→ un trigger sur la 'fille' remplaçant la FOREIGN KEY,
→ un trigger sur la 'mère' interdisant de supprimer une ligne référencée.
(4) Eventuellement les contraintes UNIQUE ou CHECK.

Ces requêtes doivent être exécutées en étant sur la base : pas de LDD distant.

5- Création de la base répartie

(a) Ecrire les requêtes permettant de créer **les objets virtuels** répartis (view ou synonym) dans les deux dictionnaires : ***Serveur1 et Serveur2***.

Un utilisateur peut se connecter sur l'une ou l'autre base et voir les objets initiaux comme si la base était centralisée sur un seul site. Reprendre exactement les mêmes noms d'objet que la base centralisée de départ.

Les objets virtuels doivent avoir la même structure et les mêmes données que les tables initiales.

Penser aussi aux Séquences.

(b) Ecrire les requêtes de consultation de ces objets virtuels sur Serveur1 et sur Serveur2

Vérifier les contenus et garder les traces.

6- Mise à jour en réparti : le commit (ou rollback) à deux phases

(a) Sur la base de ***Serveur1***, insérer deux nouveaux clients : un dans la table **Client_1** et un dans la table **Client_2** distante.

Vérifier dans chaque table la présence du nouveau client.

(b) Sur la base de ***Serveur2***, insérer deux autres clients : un dans la table **Client_2** et un dans la table **Client_1** distante.

Vérifier dans chaque table la présence du nouveau client.

(c) Sur la base de **Serveur1**, faites un **COMMIT**.

(d) Sur la base de **Serveur2**, faites un **ROLLBACK**.

Vérifier dans les deux tables des deux sites et commenter.

7- Résultats à rendre

Faire un dossier complet expliquant le déroulement du TP :

1-Objectif du TP
2-Plan
3-Progression du TP avec :
- **les ordres SQL**
- **les traces écran**
- **les remarques éventuelles**

4-Conclusion