**Université Mohamed V – Souissi Année universitaire 2009-2010**

**ENSIAS – Rabat**

**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* de la machine *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.
* CREATE TABLE Client\_1 (Nom, Prénom, Adresse) AS SELECT Nom, Prénom, Adresse FROM Client WHERE ville = 'Casablanca';
* Sur *Serveur2* : la table **Client\_2** contenant les clients de Rabat sans la colonne Ville.
* COPY FROM User11/wxcvb@ensias1 TO User11/wxcvb@ensias2 REPLACE Client\_2 (Nom, Prénom, Adresse) USING SELECT Nom, Prénom, Adresse FROM Client WHERE ville = 'Rabat';

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

* Sur *Serveur1* : la table **Compte\_1** avec les comptes appartenant aux clients de Casablanca.
* CREATE TABLE Compte\_1 (No, Type\_Compte\_No, DateOuverture, Decouvert\_autorise, Solde, Client\_No, Agence\_No) AS SELECT No, Type\_Compte\_No, DateOuverture, Decouvert\_autorise, Solde, Client\_No, Agence\_No FROM Compte WHERE Client\_No IN (SELECT No FROM Client\_1);
* Sur *Serveur2*: la table **Compte\_2** avec les comptes appartenant aux clients de Rabat.
* COPY FROM User11/wxcvb@ensias1 TO User11/wxcvb@ensias2 REPLACE TABLE Compte\_2 (No, Type\_Compte\_No, DateOuverture, Decouvert\_autorise, Solde, Client\_No, Agence\_No) USING SELECT No, Type\_Compte\_No, DateOuverture, Decouvert\_autorise, Solde, Client\_No, Agence\_No FROM Compte WHERE Client\_No IN (SELECT No FROM Client\_2);

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

* Sur *Serveur1* : la table *Operation\_1* correspondant aux opérations des comptes de la table *Compte\_1*.
* CREATE TABLE Operation\_1 (No, Type\_Operation\_No, Compte\_No) AS SELECT No, Type\_Operation\_No, Compte\_No FROM Operation WHERE Compte\_No IN (SELECT Compte\_No FROM Compte\_1)
* Sur *Serveur2* : la table *Operation\_2* correspondant aux opérations des comptes de la table *Compte\_2*.
* COPY FROM User11/wxcvb@ensias1 TO User11/wxcvb@ensias2 REPLACE Operation\_2(No, Type\_Operation\_No, Compte\_No) USING SELECT No, Type\_Operation\_No, Compte\_No FROM Operation Where Compte\_No IN (SELECT Compte\_No FROM Compte\_2)

**d-** Déplacement complet de la table **Type\_Compte** sur *Serveur2*: *Type\_Compte\_2*

COPY FROM User11/wxcvb@ensias1 TO User11/wxcvb@ensias2 REPLACE Type\_Compte\_2(no,libelle\_compte,description) USING SELECT no,libelle\_compte, description FROM Type\_Compte;

DROP TABLE Type\_Compte;

**e-** Déplacement complet de la table **Type\_Operation** sur *Serveur2* : *Type\_Operation\_2*

COPY FROM User11/wxcvb@ensias1 TO User11/wxcvb@ensias2 REPLACE Type\_Operation\_2(no, libelle\_operation, decription) USING SELECT no, libelle\_operation, description from Type\_Operation;

DROP TABLE Type\_Operation;

**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.

CREATE DATABASE LINK dbl\_ensias1 CONNECT TO User11 IDENTIFIED BY wxcvb USING 'ensias1';

CREATE DATABASE LINK dbl\_ensias2 CONNECT TO User11 IDENTIFIED BY wxcvb USING 'ensias2';

**Tester les liens établis sur chaque base en accédant aux objets distants dans les deux sens.**

CONNECT User11/wxcvb@ensias2;

SELECT \* FROM Client\_1@dbl\_ensias1;

CONNECT User11/wxcvb@ensias1;

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,

ALTER TABLE Client\_2 ADD PRIMARY KEY No;

ALTER TABLE Agence\_2 ADD PRIMARY KEY No;

ALTER TABLE Compte\_2 ADD PRIMARY KEY No;

ALTER TABLE Type\_Compte\_2 ADD PRIMARY KEY No;

ALTER TABLE Operation\_2 ADD PRIMARY KEY No;

ALTER TABLE Type\_Operation\_2 ADD PRIMARY KEY No;

1. Les contraintes de références classiques si la table ‘mère’ est sur le même site,

ALTER TABLE Compte\_2 ADD CONSTRAINT fk1 FOREIGN KEY(Client\_No) REFERENCES Client(No);

ALTER TABLE Compte\_2 ADD CONSTRAINT fk2 FOREIGN KEY(Agence\_No) REFERENCES Agence(No);

ALTER TABLE Operation\_2 ADD CONSTRAINT fk3 FOREIGN KEY(Compte\_No) REFERENCES Compte(No);

1. Les contraintes de références par ‘trigger’ si la table ‘mère’ est sur un site distant

Deux triggers :

🡪 un trigger sur la ‘fille’ remplaçant la FOREIGN KEY,

🡪 un trigger sur la ‘mère’ interdisant de supprimer une ligne référencée.

1. Eventuellement les contraintes UNIQUE ou CHECK.

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

CONNECT User11/wxcvb@ensias1;

CREATE OR REPLACE TRIGGER trig1

BEFORE DELETE OR UPDATE OF No ON Agence

FOR EACH ROW

DECLARE

x number := 0;

BEGIN

SELECT count(\*) INTO x from Compte\_2@dbl\_ensias2

WHERE Agence\_No = :old.No;

IF x>0 THEN

RAISE\_APPLICATION\_ERROR(-20175,'agence utilisée');

END IF;

END;

/

CONNECT User11/wxcvb@ensias2;

CREATE OR REPLACE TRIGGER trig2

BEFORE INSERT OR UPDATE OF Agence\_No ON Compte\_2

FOR EACH ROW

DECLARE

x number := 0;

BEGIN

SELECT count(\*) INTO x FROM Agence@dbl\_ensias1

WHERE No = :NEW.Agence\_No;

IF x=0 THEN

RAISE\_APPLICATION\_ERROR(-20175,'agence non existante');

END IF;

END;

/

CONNECT User11/wxcvb@ensias1;

CREATE OR REPLACE TRIGGER trig3

BEFORE DELETE OR UPDATE OF No ON Client

FOR EACH ROW

DECLARE

x number := 0;

BEGIN

SELECT count(\*) INTO x from Compte\_2@dbl\_ensias2

WHERE Client\_No = :old.No;

IF x>0 THEN

RAISE\_APPLICATION\_ERROR(-20175,'client utilisé');

END IF;

END;

/

CONNECT User11/wxcvb@ensias2;

CREATE OR REPLACE TRIGGER trig4

BEFORE INSERT OR UPDATE OF Client\_No ON Compte\_2

FOR EACH ROW

DECLARE

x number := 0;

BEGIN

SELECT count(\*) INTO x FROM Client@dbl\_ensias1

WHERE No = :NEW.Client\_No;

IF x=0 THEN

RAISE\_APPLICATION\_ERROR(-20175,'client non existant');

END IF;

END;

/

CONNECT User11/wxcvb@ensias1;

CREATE OR REPLACE TRIGGER trig5

BEFORE DELETE OR UPDATE OF No ON Compte

FOR EACH ROW

DECLARE

x number := 0;

BEGIN

SELECT count(\*) INTO x from Operation\_2@dbl\_ensias2

WHERE Compte\_No = :old.No;

IF x>0 THEN

RAISE\_APPLICATION\_ERROR(-20175,'compte utilisé);

END IF;

END;

/

CONNECT User11/wxcvb@ensias2;

CREATE OR REPLACE TRIGGER trig6

BEFORE INSERT OR UPDATE OF Compte\_No ON Operation\_2

FOR EACH ROW

DECLARE

x number := 0;

BEGIN

SELECT count(\*) INTO x FROM Compte@dbl\_ensias1

WHERE No = :NEW.Compte\_No;

IF x=0 THEN

RAISE\_APPLICATION\_ERROR(-20175,'compte non existant');

END IF;

END;

/**5- Création de la base répartie**

1. 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.

CONNECT User11/wxcvb@ensias1;

CREATE VIEW Client (no,Nom,Prenom,Adresse,Ville) AS

SELECT no,Nom,Prenom,Adresse,'Casablanca' FROM Client\_1

UNION

SELECT no,Nom,Prenom,Adresse,'Rabat' FROM Client\_2@db1\_ensias2;

CONNECT User11/wxcvb@ensias2;

CREATE VIEW Client (no,Nom,Prenom,Adresse,Ville) AS

SELECT no,Nom,Prenom,Adresse,'Casablanca' FROM Client\_1@db1\_ensias1

UNION

SELECT no,Nom,Prenom,Adresse,'Rabat' FROM Client\_2;

1. **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**

1. Sur la base de ***Serveur1***,insérer deux nouveaux clients : un dans la table **Client\_1** et un dans la table **Client\_2** distante.

INSERT INTO Client\_1 VALUES('TEBBAI', 'AYOUB', 'Hay Nahda');

INSERT INTO Client\_2@dbl\_Ensias2 VALUES('BOUGHTI', 'HOUSSAM', 'Hay Saada');

Vérifier dans chaque table la présence du nouveau client.

SELECT \* FROM Client\_1;

SELECT \* FROM Client\_2@dbl\_Ensias2;

1. Sur la base de ***Serveur2***, insérer deux autres clients : un dans la table **Client\_2** et un dans la table **Client\_1** distante.

INSERT INTO Client\_2 VALUES('BAHIDA', 'MERIEME', 'Hay Nahda'

INSERT INTO Client\_1@dbl\_Ensias1 VALUES('BAHIDA', 'AYA', 'Hay Saada');

Vérifier dans chaque table la présence du nouveau client.

SELECT \* FROM Client\_2;

SELECT \* FROM Client\_1@dbl\_Ensias1;

1. Sur la base de **Serveur1**, faites un **COMMIT**.

COMMIT;

1. Sur la base de **Serveur2**, faites un **ROLLBACK**.

ROLLBACK;

Vérifier dans les deux tables des deux sites et commenter.

Les 4 lignes n'étaient pas insérées à cause du ROLLBACK.