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;
```

(2) 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);
```

(3) 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.

(4) 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

(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.

```
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;
```

(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.

```
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;
```

(b) 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;
```

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

COMMIT;

(d) 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.