Deuxième année cycle d'ingénieur
- GI -
2022/2023

Base de données Avances

Répartition de base de données

BANQUE

Réalisé par :

Nouhaila AZLAG

Chaimaa FAKHAM

I. Introduction :

Les bases de données distribuées peuvent prendre différentes formes en fonction de l'architecture et des mécanismes utilisés pour répartir les données et les requêtes. Certaines architectures couramment utilisées comprennent les bases de données distribuées horizontalement, où les données sont réparties entre plusieurs nœuds de manière égale, et les bases de données distribuées verticalement, où les données sont divisées en fonction des attributs ou des colonnes.

Les bases de données distribuées sont utilisées dans de nombreux domaines, tels que les entreprises, les systèmes de gestion de contenu, les réseaux sociaux et les applications web à grande échelle. Elles offrent des avantages significatifs en termes de performance et de disponibilité, mais elles peuvent également poser des défis supplémentaires en matière de cohérence des données, de gestion de la réplication et de la synchronisation entre les nœuds.

Définition :

Une base de données distribuée ou BDD est une base de données dont la gestion est traitée par un réseau d'ordinateur interconnectes qui stockent des données de manière distribuée.

Les avantages de la répartition :

Une base de données distribuée offre une solution puissante pour le stockage et la gestion des données à grande échelle, parmi ses avantages :

- ✓ Il devient impératif de **décentraliser l'information** (cas des multinationales),
- ✓ Augmentation du volume de Données.
- ✓ Augmentation du volume des transactions.

Travail à Faire :

Notre banque possède des succursales dans différentes régions géographiques. Plutôt que de stocker toutes les données sur un seul serveur centralisé, elle peut utiliser une base de données distribuée pour répartir les données entre les différentes succursales. Cela permet à chaque succursale d'avoir accès aux informations locales tout en maintenant une vue globale cohérente des données à l'échelle de l'ensemble de la banque.

En répartissant les données sur plusieurs nœuds, la banque peut accélérer les opérations de traitement et de recherche. Les requêtes peuvent être exécutées en parallèle, réduisant les temps de réponse et améliorant l'efficacité globale du système.

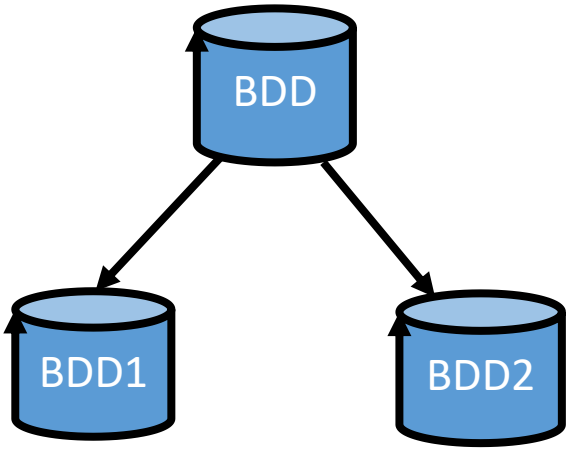
SITE1 : (BBD1)

- $R1 = \sigma\ VilleClient = 'Casablanca'\ AND\ Solde < 0(Client \bowtie Compte)$

SITE2 : (BBD2)

- $R2 = \sigma_{VilleClient='Rabat'\ AND\ Solde \geq 0}\ (Client \bowtie Compte).$

Notre objectif est de gérer les clients et les comptes dans la base des données. On base sur la relation **Down up.**



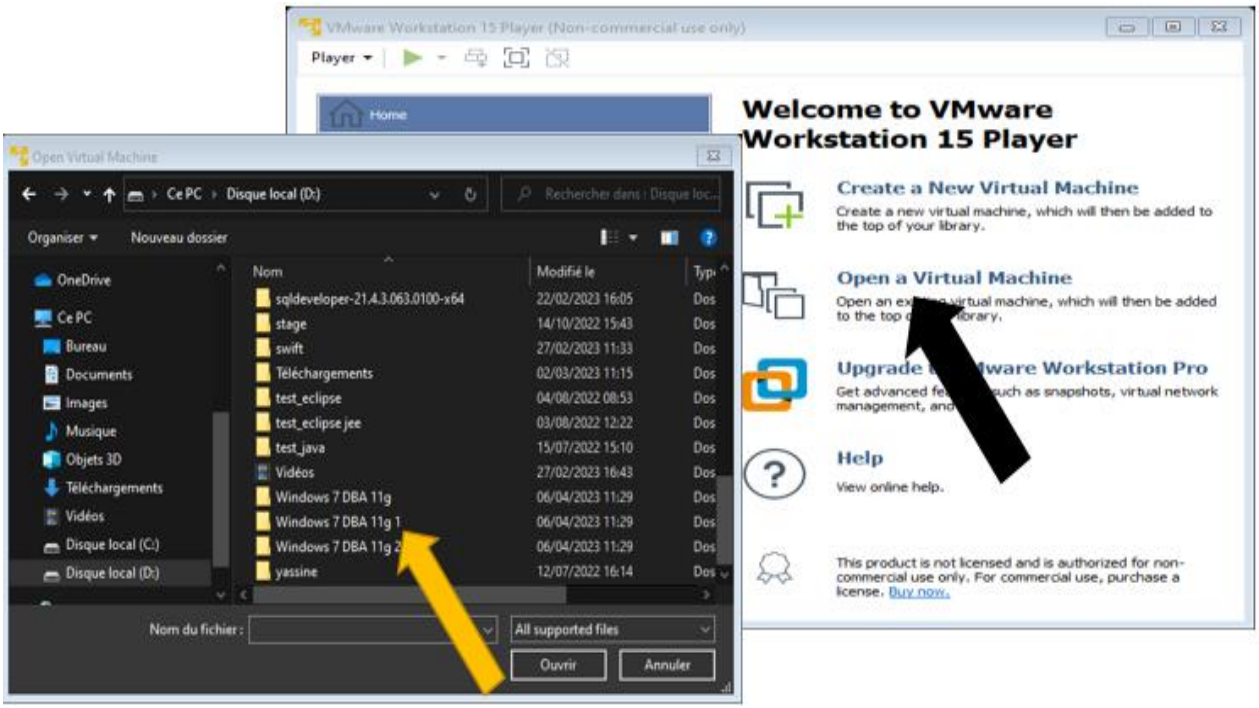
Etapes de création

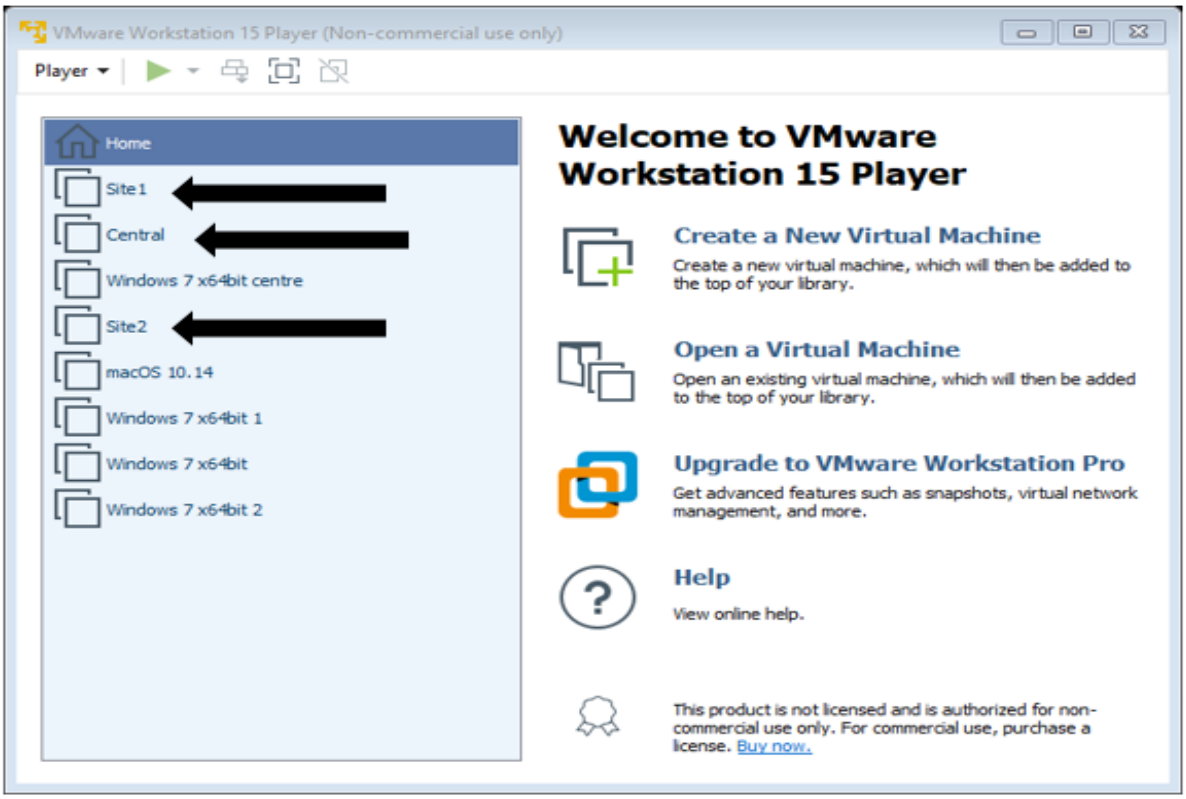
- Création d'un réseau comportant 3 sites : Le central , le site1 et le site2 .
- Création de nouveaux comptes : userCentre , userCasa et userRabat .
- Création les liens des bases de données : lienCentre , lienCasa et lienRabat .
- Création de la base de données centrale : BANK .
- Création des fragments : Bank.BDD1 et Bank.BDD2 .
- Création des synonymes .
- Synchronisation des insertions , des suppression et des mises à jour .

II. Création d'un réseau virtuel :

- Télécharger et installer VMware .
- Télécharger une image Windows 7 .
- Installer oracle 11g et SQL développeur dans la nouvelle machine.

Création des autres machines



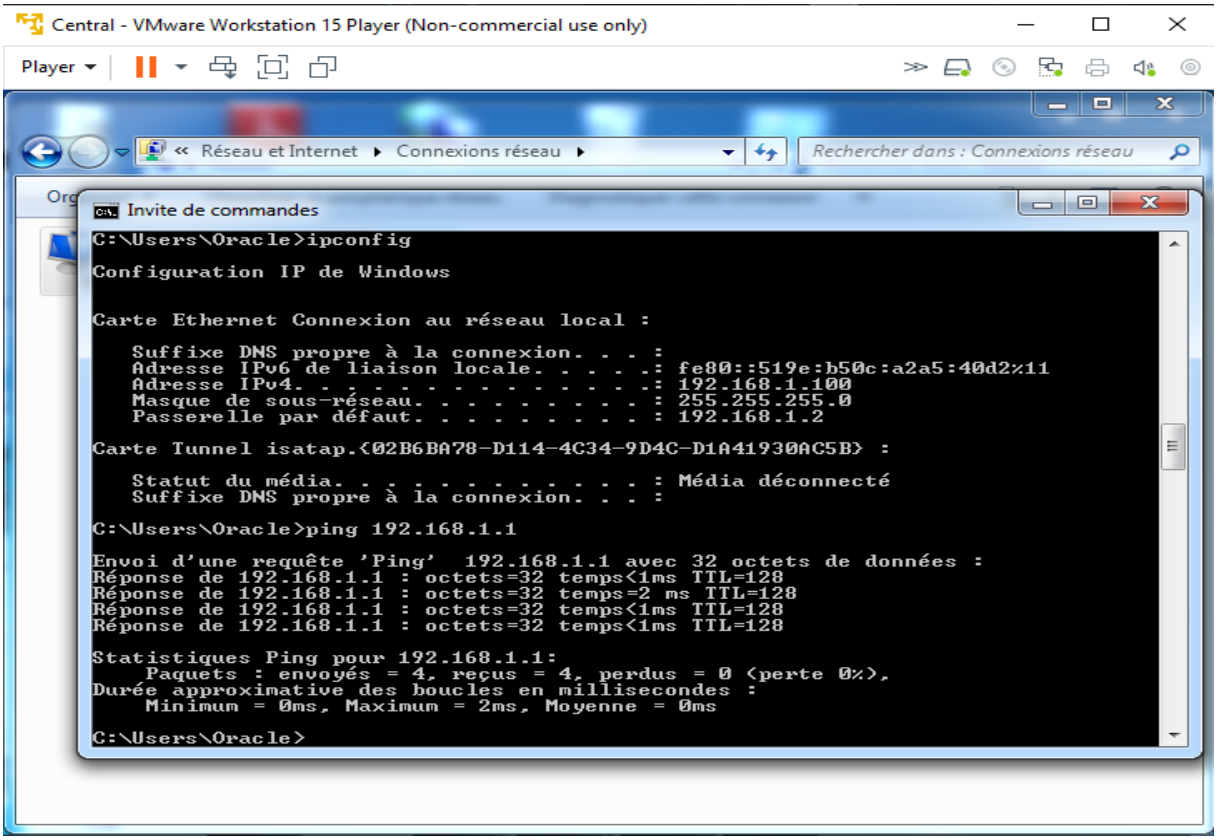


Configuration de réseaux de machines virtuelle :

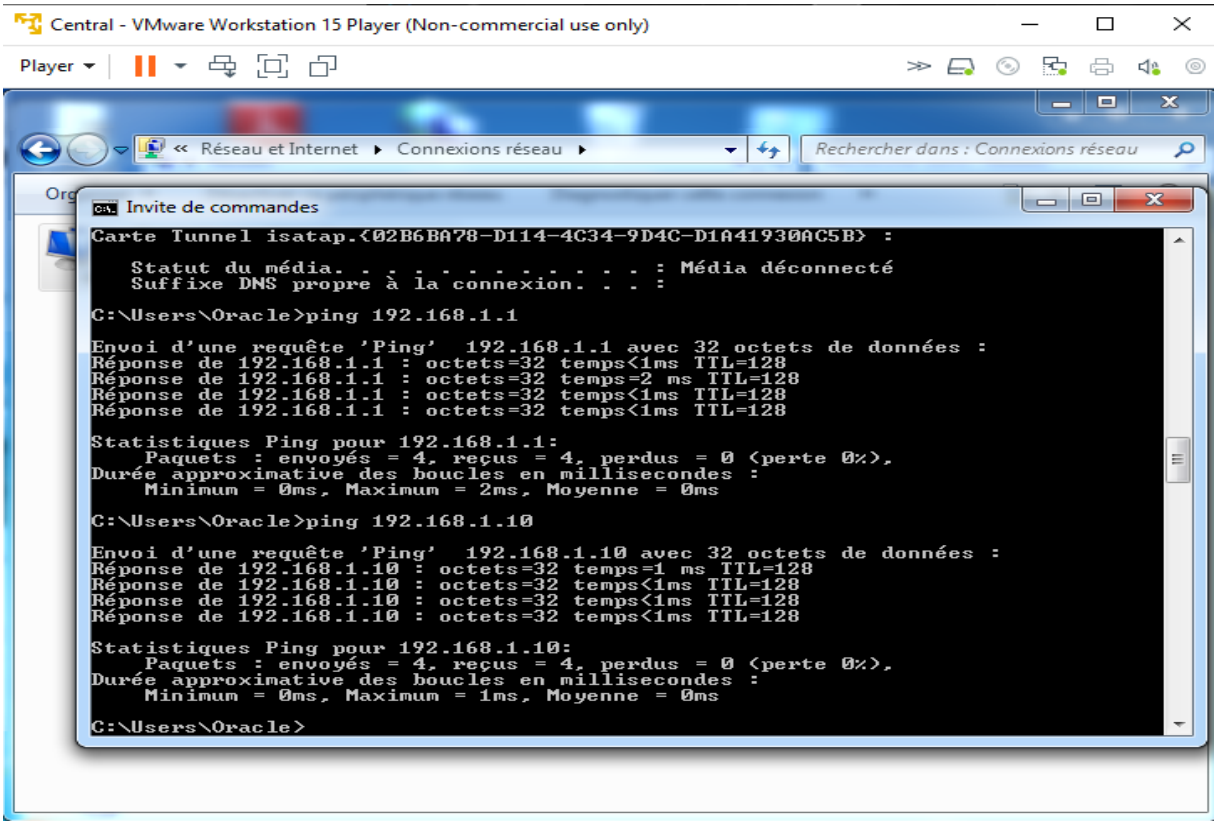
Choisir un MV parmi les trois machines crée et entrer dans le centre de partage de réseaux pour affecter IP adresse à la machine, et on fait la même chose pour les autres MV. les adresses IP des MV doit tout sur la même classe pour on peut connecter.

- BDD central : 192.168.1.100
- BDD Casablanca : 192.168.1.1
- BDD Rabat :192.168.1.10

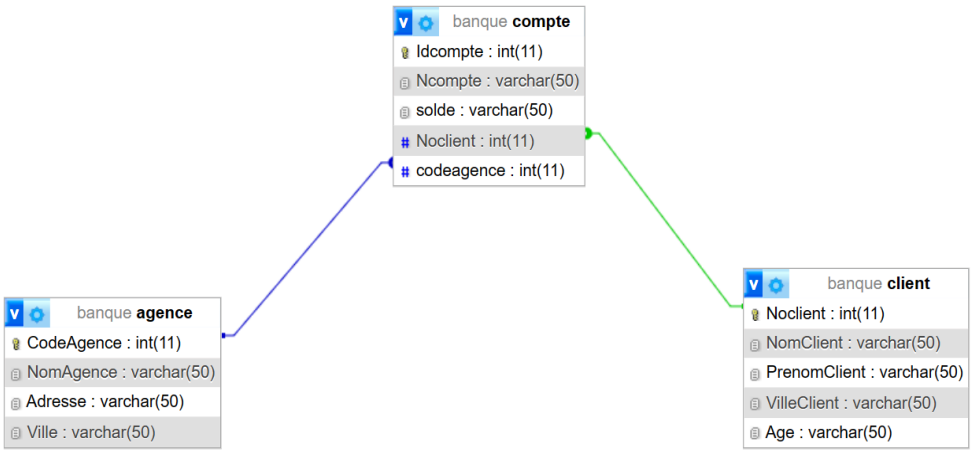
testez la connectivité entre les machines virtuels.
ping entre central et site1:



ping entre central et site2:



Modelé logique de donnée :



Création des nouveaux comptes:

- La Banque central :

```
CREATE USER Central IDENTIFIED BY Central
GRANT ALL PRIVILEGES TO Central
```

- La Banque Site 1 :

```
CREATE USER siteCasa IDENTIFIED BY Site1
GRANT ALL PRIVILEGES TO siteCasa
```

- La Banque Site 2:

```
CREATE USER userSite2 IDENTIFIED BY Site2
GRANT ALL PRIVILEGES TO userSite2.
```

- Pour Vérifier la création des utilisateurs:

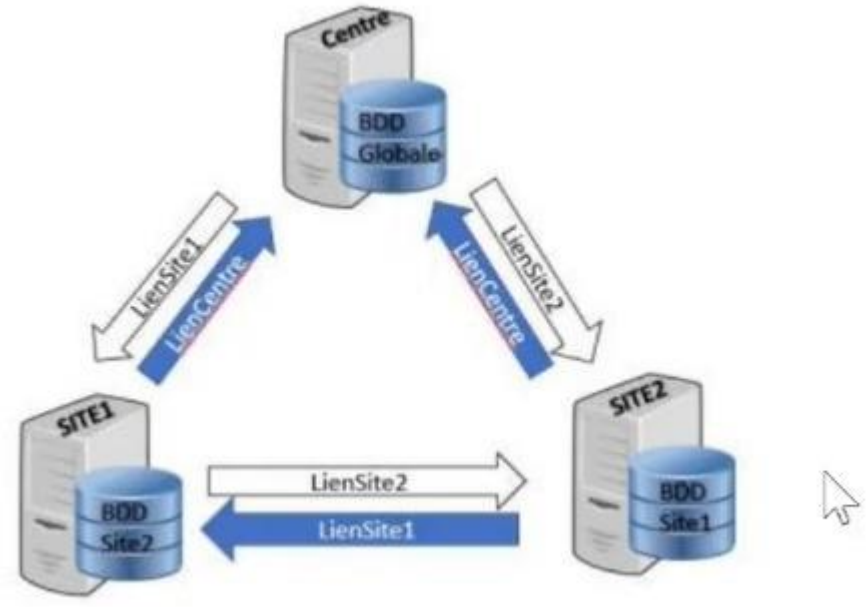
```
select username  from DBA_USERS.
```

Création des database link :

- Dans la banque central on va créer un lien vers les deux sites :

1.
```
CREATE PUBLIC DATABASE LINK linkCasa
  CONNECT TO siteCasa IDENTIFIED BY siteCasa
  USING ' siteCasa:1552/ORCL';
```

2.
```
CREATE PUBLIC DATABASE LINK lienSite2
  CONNECT TO userSite2 IDENTIFIED BY Site2
  USING 'Site2:1552/ORCL';
```



Pour tester le bon fonctionnement des liens:

SELECT sysdate FROM dual@NOMlien

Création des synonymes des tables:

Pour Casablanca :

CREATE PUBLIC SYNONYM Client1 FOR ClientSite1@ linkCasa;

CREATE PUBLIC SYNONYM Compte1 FOR CompteSite1@ linkCasa;

CREATE PUBLIC SYNONYM Agence1FOR AgenceSite1@ linkCasa;

Même démarche pour le site 2,3.

Création des tables:

- Central :

```
CREATE TABLE Client (
  Noclient number default NULL,
  NomClient varchar2(255) default NULL,
  PrenomClient varchar2(255) default NULL,
  VilleClient varchar2(255) default NULL,
  Age varchar2(50) default NULL
);
CREATE TABLE Agence (
  CodeAgence number default NULL,
  NomAgence varchar2(255),
  Adresse varchar2(255) default NULL,
  Ville varchar2(255) default NULL
);
CREATE TABLE Compte (
  Idcompte number default NULL,
  NComte varchar2(11) default NULL,
  solde varchar2(50) default NULL,
  Noclient varchar2(50) default NULL,
  CodeAgence varchar2(50) default NULL
);

ALTER TABLE Client add Constraint c1 Primary Key (Noclient);
ALTER TABLE Compte add Constraint c2 Primary Key (Idcompte);
ALTER TABLE Compte add Constraint f1 Foreign Key (Noclient) References Client(Noclient) on DELETE Cascade;
```

❖ Casablanca :

```
CREATE TABLE Compte1 as (SELECT Compte.* FROM Client@lienc_centre, Compte@lienc_centre WHERE Client.Noclient=Compte.Noclient
AND VilleClient='Casablanca' AND Solde<0);
CREATE TABLE Client1 as (SELECT Distinct Client.* FROM Client@lienc_centre, Compte1 WHERE Client.Noclient=Compte1.Noclient) ;
ALTER TABLE Client1 add Constraint c1 Primary Key (Noclient);
ALTER TABLE Compte1 add Constraint c2 Primary Key (Idcompte);
ALTER TABLE Compte1 add Constraint f1 Foreign Key (Noclient) References Client1(Noclient) on DELETE Cascade;
```

❖ Rabat :

```
0,0015311 seconds

CREATE TABLE Compte2 as (SELECT Compte.* FROM Client@liens_centre, Compte@liens_centre WHERE Client.Noclient=Compte.Noclient
AND VilleClient='Rabat' AND Solde>=0);
CREATE TABLE Client2 as (SELECT Distinct Client.* FROM Client@lienc_centre, Compte2 WHERE Client.Noclient=Compte1.Noclient) ;
ALTER TABLE Client2 add Constraint c1 Primary Key (Noclient);
ALTER TABLE Compte2 add Constraint c2 Primary Key (Idcompte);
ALTER TABLE Compte2 add Constraint f1 Foreign Key (Noclient) References Client2(Noclient) on DELETE Cascade;
```

Création des Procédures:

➢ **Insertion des Clients :** A partir de base de données centrale

```
create or replace PROCEDURE REPCLIENT IS
CURSOR C IS
SELECT * FROM CLIENT@LIENC_CENTRE
WHERE NoClient IN (SELECT NoClient FROM  Compte1);

TYPE rec IS RECORD (param Client@LIENC_CENTRE%ROWTYPE);
R rec;

BEGIN
OPEN C;
FETCH C INTO R.param;
WHILE c%FOUND LOOP
INSERT INTO Client1 VALUES R.param;
FETCH C INTO R.param;
END LOOP;
COMMIT;
END;
```

➢ **Insertion des Comptes :** A partir de base de données centrale dans les deux sites Casablanca et Rabat.

❖ Casablanca :

```
create or replace PROCEDURE REPCOMPTE IS
  CURSOR C IS
    SELECT COMPTE.*
    FROM Client@LIENC_CENTRE, Compte@LIENC_CENTRE
    WHERE Client.NoClient@LIENC_CENTRE = Compte.NoClient@LIENC_CENTRE
    AND VilleClient='Casablanca' AND Solde<0;

  TYPE rec IS RECORD (param Compte1%ROWTYPE);
  R rec;
BEGIN
  OPEN C;
  FETCH C INTO R.param;

  WHILE C%FOUND LOOP
    INSERT INTO Compte1 VALUES R.param;
    FETCH C INTO R.param;
  END LOOP;

  COMMIT;
END;
```

❖ Rabat :

```
create or replace PROCEDURE REPCOMPTE IS
  CURSOR C IS
    SELECT COMPTE.*
    FROM Client@LIENS_CENTRE, Compte@LIENS_CENTRE
    WHERE Client.NoClient@LIENS_CENTRE = Compte.NoClient@LIENS_CENTRE
    AND VilleClient='Rabat' AND Solde>=0;
  TYPE rec IS RECORD (param Compte2%ROWTYPE);
  R rec;
BEGIN
  OPEN C;
  FETCH C INTO R.param;

  WHILE C%FOUND LOOP
    INSERT INTO Compte2 VALUES R.param;
    FETCH C INTO R.param;
  END LOOP;
  COMMIT;
END;
```

Création des Triggers:

Synchronisation de la répartition de l'insertion, de la suppression des mises à jour

Insertion des comptes :

Central vers Casablanca:

```
create or replace TRIGGER INSERT_COMPTE
BEFORE INSERT ON compte
FOR EACH ROW
DECLARE
  V client.villeclient%TYPE;
  S compte.solde%TYPE := :new.solde;
  P NUMBER;
  TYPE str IS RECORD (L client%ROWTYPE);
  E str;
BEGIN
  SELECT villeclient INTO V
  FROM client
  WHERE client.noclient = :new.noclient;

  IF (V = 'Casablanca' AND S < 0) THEN
    INSERT INTO compte1@LIENCASA VALUES (:new.idcompte, :new.ncomte, :new.solde, :new.noclient, :new.codeAgence);
  ELSIF (V = 'Rabat' AND S > 0) THEN
    INSERT INTO compte2@LIENRABAT VALUES (:new.idcompte, :new.ncomte, :new.solde, :new.noclient, :new.codeAgence);
  END IF;
END;
```

Supprimer un Compte :

```
create or replace TRIGGER delete_compte
AFTER DELETE ON compte
FOR EACH ROW
begin
DECLARE
  V client.villeclient%TYPE;
  S compte.solde%TYPE := :old.solde;
BEGIN
  SELECT villeclient INTO V FROM client WHERE client.Noclient = :old.Noclient;

  IF (V = 'Casablanca' AND S < 0) THEN
    DELETE FROM compte1@liencasa WHERE idcompte=:old.idcompte;
  ELSIF (V = 'Rabat' AND S >=0) THEN
    DELETE FROM compte2@lienrabat WHERE idcompte=:old.idcompte;
  END IF;
end;
END;
```

Modifier un Compte :

```
create or replace TRIGGER ModifierCompte
AFTER UPDATE ON compte
FOR EACH ROW
BEGIN
  DECLARE
    NVille client.villeclient%TYPE;
    OVille client.villeclient%TYPE;
    NSolde compte.solde%TYPE := :new.solde;
    OSolde compte.solde%TYPE := :old.solde;
  BEGIN
    SELECT villeclient INTO NVille
    FROM client
    WHERE client.noclient = :new.noclient;
    SELECT villeclient INTO OVille
    FROM client
    WHERE client.noclient = :old.noclient;
    IF (OVille = 'Casablanca' AND OSolde < 0) THEN
      IF (NVille = 'Casablanca' AND NSolde < 0) THEN
        UPDATE compte1@liencasa
        SET solde = :new.solde
        WHERE idcompte = :new.idcompte;
      ELSE
        DELETE FROM compte1@liencasa WHERE idcompte = :new.idcompte;
        IF (NVille = 'Rabat' AND NSolde > 0) THEN
          INSERT INTO compte2@lienrabat VALUES (:new.idcompte, :new.ncomte, :new.solde, :new.noclient, :new.codeAgence);
```

```
      ELSE
        DELETE FROM compte1@liencasa WHERE idcompte = :new.idcompte;
        IF (NVille = 'Rabat' AND NSolde > 0) THEN
          INSERT INTO compte2@lienrabat VALUES (:new.idcompte, :new.ncomte, :new.solde, :new.noclient, :new.codeAgence);
        END IF;
      END IF;
    ELSIF (OVille = 'Rabat' AND OSolde > 0) THEN
      IF (NVille = 'Rabat' AND NSolde > 0) THEN
        UPDATE compte2@lienrabat
        SET solde = :new.solde
        WHERE idcompte = :new.idcompte;
      ELSE
        DELETE FROM compte2@lienrabat WHERE idcompte = :new.idcompte;

        IF (NVille = 'Casablanca' AND NSolde < 0) THEN
          INSERT INTO compte1@liencasa VALUES (:new.idcompte, :new.ncomte, :new.solde, :new.noclient, :new.codeAgence);
        END IF;
      END IF;
    ELSIF (NVille = 'Casablanca' AND NSolde < 0) THEN
      INSERT INTO compte1@liencasa VALUES (:new.idcompte, :new.ncomte, :new.solde, :new.noclient, :new.codeAgence);
    ELSIF (NVille = 'Rabat' AND NSolde > 0) THEN
      INSERT INTO compte2@lienrabat VALUES (:new.idcompte, :new.ncomte, :new.solde, :new.noclient, :new.codeAgence);
    END IF;
  END;
END;
```

Insertions des clients dans les deux sites (Casablanca et Rabat)

❖ Casablanca :

```
create or replace TRIGGER Insert_New_clients
BEFORE INSERT ON compte1
FOR EACH ROW
DECLARE
  p NUMBER;
  TYPE str IS RECORD(l client1%ROWTYPE);
  E str;
BEGIN
  BEGIN
    SELECT NoClient INTO p FROM Client1 WHERE NoClient = :new.NoClient;
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      SELECT * INTO E.L FROM Client1 WHERE Client1.NoClient = :new.NoClient;
      INSERT INTO Client1 VALUES E.L;
  END;
END;
```

❖ Rabat : Même principe.

La suppression des clients dans les deux sites après la suppression de compte

❖ Casablanca :

```
create or replace TRIGGER delete_client
AFTER DELETE ON compte1
FOR EACH ROW
DECLARE
  P1 NUMBER;
  P2 NUMBER;
BEGIN
  BEGIN
    SELECT COUNT(DISTINCT Noclient) INTO P2
    FROM compte1
    WHERE Noclient = :old.Noclient
      AND Solde >= 0;
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
      DELETE FROM client1 WHERE Noclient = :old.Noclient;
  END;
END;
```

❖ Rabat : Même principe.

Conclusion :

La base de données distribuée reste une solution attrayante pour les entreprises qui nécessitent une grande évolutivité, une haute disponibilité et une résilience accrue. Elle offre la possibilité de traiter de grandes quantités de données de manière efficace, tout en assurant une meilleure performance et une meilleure protection des données.