Compte-rendu SGBD TP n°3

Charles Javerliat, Pierre Sibut-Bourde
Aymen Ould Hamouda, Yann Lafaille
Timothé Berthier, Paul Grévaud
INSA Lyon, 3-IF

21 janvier 2020

Résumé

Ce TP traite de la mise en œuvre d'une distribtion d'une base de donnée pour une entreprise nommée `Ryori` et opérante sur trois grandes régions du monde : Amérique, Europe du Sud, Europe du Nord. Ce sujet propose la mise en place de quatre applications déployées `MakeIt` (fabrication), `DesignIt` (conception), `SellIt` (vente) et `RH` (ressources humaines).

Table des matières

I Contexte et objectifs

Ce TP traite de la mise en œuvre d'une distribtion d'une base de donnée pour une entreprise nommée `Ryori` et opérante sur trois grandes régions du monde (ici, distribution sur trois binômes) : Amérique, Europe du Sud, Europe du Nord. Ce sujet propose la mise en place de quatre applications déployées `MakeIt` (fabrication), `DesignIt` (conception), `SellIt` (vente) et `RH` (ressources humaines). L'objectif majeur est d'opérer cette conversion de telle sorte que rien ne soit perdu dans l'opération : ni les possibilités de la table originelle, ni aucun tuples dans une requête quelconque dans la base de donnée (une requête `SFW` doit rendre la même chose dans la base centralisée et dans la nouvelle distribuée).

II Présentation rôles

II.A Groupes

— Charles Javerliat, Pierre Sibut-Bourde `B3109`
— Aymen Ould Hamouda, Yann Lafaille `B3111`
— Timothé Berthier, Paul Grévaud `B3101`

II.B Tâches et répartition

Amérique (Am) : `B3101`
Europe du Nord (EN) : `B3109`
Europe du Sud (ES) : `3111`
Responsables :

— Coordination générale : Charles Javerliat ∈ `B3109`.
— Documentation et Rapport à rendre : Pierre Sibut-Bourde ∈ `B3109`.

II.C Code

On évitera, dans ce compte-rendu, au maximum les redites. Cependant, travailler sur le code commenté suppose autant que possible de pouvoir embrasser d'un même regard et le code, et le commentaire. On utilisera donc des fragments de codes, mais pour tout binôme, on sait que : **on trouvera pour toute partie le code complet dans la partie correspondante »**

III Fragmentation

III.A Détermination des fragments

III.A.1 Table Stock (fragmentation horizontale)

On lit dans le sujet que l'on peut séparer le stock global en fonction de la région considérée, avec un cas particulier pour l'Allemagne pour l'application `MakeIt` développée par l'Europe du Nord. On écrit la partition de Stock comme :

$$\text{Stock} = \underbrace{(\text{Stock}_{\text{Allemagne}} \sqcup \text{Stock}_{\text{EUN not All.}} \sqcup \text{Stock}_{\text{Autre}})}_{\text{Europe du Nord}} \sqcup \underbrace{\text{Stock}_{\text{EUS}}}_{\text{Europe du Sud}} \sqcup \underbrace{\text{Stock}_{\text{Am}}}_{\text{Amérique}}$$

En utilisant cela, on peut fragmenter horizontalement ($\sqcup$ indique l'union disjointe) la table de stock pour que la gestion se fasse en des sites différents. Ce qui en relations algébriques donnera :

$$Stock_{Allemagne} = \sigma_{(Pays=Allemagne)}(Stock)$$

$$Stock_{EUN} = \sigma_{(Pays \in PaysEUN) \wedge (Pays \neq Allemagne)}(Stock)$$

$$Stock_{EUS} = \sigma_{(Pays \in PaysEUS)}(Stock)$$

$$Stock_{Am} = \sigma_{(Pays \in PaysAm)}(Stock)$$

$$Stock_{Autre} = \sigma_{(Pays \notin (PaysEUN \cup PaysEUS \cup PaysAm))}(Stock)$$

Justification : L'application `MakeIt`, développée en Europe du Nord, a l'accès total au stock allemand, donc on crée un fragment correspondant. Mais elle utilise également le stock local pour l'application `SellIt` et,

pour éviter le doublon, on peut créer une table locale excluant la table allemande. De plus, pour l'ouverture internationale (marchés asiatique, africain, océanien) il faut la gestion des autres pays et, comme le siège social est situé en Allemagne, on s'occupe du stock autre dans cette partie. La gestion locale de l'Europe du Sud et de l'Amérique se justifie par l'application `SellIt`.

III.A.2 Table Clients (fragmentation horizontale)

On lit dans le sujet que l'on peut séparer les clients en fonction de la région considérée. On détermine alors quatre fragments associés à la table Clients utilisée dans l'application `SellIt` afin de fragmenter horizontalement selon la partition suivante. La gestion des clients autres se fait sur la base de donnée associée à l'Europe du Nord (cf. ci-dessus avec l'ouverture internationale)

$$\text{Clients} = \underbrace{(\text{Clients}_{\text{EUN}} \sqcup \text{Clients}_{\text{Autre}})}_{\text{Europe du Nord}} \sqcup \underbrace{\text{Clients}_{\text{EUS}}}_{\text{Europe du Sud}} \sqcup \underbrace{\text{Clients}_{\text{Am}}}_{\text{Amérique}}$$

$$Clients_{EUN} = \sigma_{(Pays \in PaysEUN)}(Clients)$$
$$Clients_{EUS} = \sigma_{(Pays \in PaysEUS)}(Clients)$$
$$Clients_{Am} = \sigma_{(Pays \in PaysAm)}(Clients)$$
$$Clients_{Autre} = \sigma_{(Pays \notin (PaysEUN \cup PaysEUS \cup PaysAm))}(Clients)$$

III.A.3 Table Commandes (fragmentation horizontale dérivée)

Pour la table Commandes utilisée dans l'application `SellIt`, on réalise une fragmentation horizontale dérivée avec une selection par semi-jointure entre les commandes et les clients, ainsi chaque base de donnée possédera les commandes de ses propres clients :

$$Commandes_{EUN} = Commandes \ltimes Clients_{EUN}$$

$$Commandes_{EUS} = Commandes \ltimes Clients_{EUS}$$
$$Commandes_{Am} = Commandes \ltimes Clients_{Am}$$

Afin de récupérer seulement les commandes relatives à des clients pour une région donnée. Cela donne quatre fragments, équivalents à la partie précédente :

$$\text{Commande} = \underbrace{(\text{Commande}_{\text{EUN}} \sqcup \text{Commande}_{\text{Autre}})}_{\text{Europe du Nord}} \sqcup \underbrace{\text{Commande}_{\text{EUS}}}_{\text{Europe du Sud}} \sqcup \underbrace{\text{Commande}_{\text{Am}}}_{\text{Amérique}}$$

III.A.4 Table Details_Commandes (fragmentation horizontale dérivée)

Pour la table DetailsCommandes utilisée dans l'application `SellIt`, on réalise une fragmentation horizontale dérivée avec une selection par semi-jointure entre les détails des commandes et les commandes, afin de récupérer seulement le détail des commandes relatives aux commandes présentes dans la base :

$$Details\_Commandes_{EUN} = Details\_Commandes \ltimes Commandes_{EUN}$$

$$Details\_Commandes_{EUS} = Details\_Commandes \ltimes Commandes_{EUS}$$
$$Details\_Commandes_{Am} = Details\_Commandes \ltimes Commandes_{Am}$$

III.A.5 Autres tables

Pour les tables Employés, Fournisseurs, Catégories ou Produits, on un seul fragment que l'on situera respectivement dans les bases de données gérées par le site Amérique, Europe du Nord, Europe du Sud (deux fois).

III.B Placement des fragments sur les sites (sans réplication)

III.B.1 Analyse

Afin de placer les fragments sur les sites, il faut effectivement fragmenter la table initiale comme indiqué précédemment, ce qui permet d'être plus efficace, en cela que chaque site gère la base à laquelle il accède le plus souvent.

III.B.2 Bilan général de la fragmentation

Le bilan est alors, site par site :

```
Europe du Nord :
   Clients Autres
   Clients EU N
   Commandes Autres
   Commandes EU N
   Detail Commandes Autres
   Detail Commandes EU N
   Fournisseurs
   Stock Allemagne
   Stock Autres
   Stock EU N
Europe du Sud :
   Categories
   Clients EU S
   Commandes EU S
   Detail Commandes EU S
   Produits
   Stock EU S
Amerique :
   Clients AM
   Commandes AM
   Detail Commandes AM
   Employes
   Stock AM
```

III.C Mise en œuvre de la base sans réplication

III.C.1 Site Europe du Nord

Binôme responsable Javerliat Charles, Sibut-Bourde Pierre, binôme numéro B3109

Connexion à la base centralisée

```
CREATE DATABASE LINK DBLCentrale
CONNECT TO cjaverliat
IDENTIFIED BY ****
USING 'DB11';

-- On peut tester le bon établissement de la connexion avec une requête SFW
select * from Ryori.clients@DBLCentrale;
```

Création des liens entre les bases Pour communiquer avec les autres bases de données (Amérique et Europe du Sud), on a besoin de créer des liens.

```
CREATE DATABASE LINK DBEuropeSud
CONNECT TO cjaverliat
IDENTIFIED BY ****
USING 'DB13';

CREATE DATABASE LINK DBAmerique
CONNECT TO cjaverliat
IDENTIFIED BY ****
USING 'DB14';

-- On peut tester le bon établissement des connexions avec une requête SFW
select * from aouldhamou.clients_eu_s@DBEuropeSud;
select * from tberthier.clients_am@DBAmerique;
```

Création et peuplement des tables

```
--Migration de Fournisseurs
CREATE TABLE Fournisseurs AS (SELECT * FROM Ryori.fournisseurs@DBLCentrale);

--Stock de l'Europe du Nord hors Allemagne
CREATE TABLE Stock_EU_N AS (
SELECT * FROM Ryori.Stock@DBLCentrale WHERE Pays IN (
  'Norvege','Suede','Danemark','Islande','Finlande','Royaume-Uni','Irlande',
  'Belgique','Luxembourg','Pays-Bas','Pologne'
  ));

--Stock de l'Allemagne
CREATE TABLE Stock_Allemagne AS (
SELECT * FROM Ryori.Stock@DBLCentrale WHERE Pays = 'Allemagne'
);

--Stock des autres regions que Eur. N., Eur. S., Amerique
CREATE TABLE Stock_Autres AS (
SELECT * FROM Ryori.Stock@DBLCentrale WHERE Pays NOT IN (
  --Europe du Nord
  'Norvege', 'Suede', 'Danemark', 'Islande', 'Finlande', 'Royaume-Uni',
  'Irlande', 'Belgique', 'Luxembourg', 'Pays-Bas', 'Pologne', 'Allemagne',
  --Europe du Sud
  'Espagne', 'Portugal', 'Andorre', 'France', 'Gibraltar', 'Italie',
  'Saint-Marin', 'Vatican', 'Malte', 'Albanie', 'Bosnie-Herzegovine',
  'Croatie', 'Grece', 'Macedoine',
  'Montenegro', 'Serbie', 'Slovenie', 'Bulgarie',
  --Amerique
  'Antigua-et-Barbuda', 'Argentine', 'Bahamas', 'Barbade', 'Belize',
  'Bolivie', 'Bresil', 'Canada', 'Chili', 'Colombie', 'Costa Rica', 'Cuba',
  'Republique dominicaine', 'Dominique',
  'Equateur', 'Etats-Unis', 'Grenade', 'Guatemala', 'Guyana', 'Haiti',
  'Honduras', 'Jamaique',
  'Mexique', 'Nicaragua', 'Panama', 'Paraguay', 'Perou',
  'Saint-Christophe-et-Nieves', 'Sainte-Lucie',
  'Saint-Vincent-et-les Grenadines', 'Salvador', 'Suriname',
  'Trinite-et-Tobago', 'Uruguay', 'Venezuela'
  ));

--Clients de l'Europe du Nord
CREATE TABLE Clients_EU_N AS(
SELECT * FROM Ryori.Clients@DBLCentrale WHERE Pays IN (
  'Norvege','Suede','Danemark','Islande','Finlande','Royaume-Uni',
  'Irlande','Belgique','Luxembourg','Pays-Bas','Pologne', 'Allemagne'
));

--Clients des autres regions que Eur. N., Eur. S., Amerique
CREATE TABLE Clients_Autres AS(
SELECT * FROM Ryori.Clients@DBLCentrale WHERE Pays NOT IN (
  --Europe du Nord
  'Norvege', 'Suede', 'Danemark', 'Islande', 'Finlande', 'Royaume-Uni',
  'Irlande', 'Belgique', 'Luxembourg', 'Pays-Bas', 'Pologne', 'Allemagne',
  --Europe du Sud
  'Espagne', 'Portugal', 'Andorre', 'France', 'Gibraltar', 'Italie',
  'Saint-Marin', 'Vatican', 'Malte', 'Albanie', 'Bosnie-Herzegovine',
  'Croatie', 'Grece', 'Macedoine',
  'Montenegro', 'Serbie', 'Slovenie', 'Bulgarie',
  --Amerique
  'Antigua-et-Barbuda', 'Argentine', 'Bahamas', 'Barbade', 'Belize',
  'Bolivie', 'Bresil', 'Canada', 'Chili', 'Colombie', 'Costa Rica',
```

```
  'Cuba', 'Republique dominicaine', 'Dominique',
  'Equateur', 'Etats-Unis', 'Grenade', 'Guatemala', 'Guyana',
  'Haiti', 'Honduras', 'Jamaique',
  'Mexique', 'Nicaragua', 'Panama', 'Paraguay', 'Perou',
  'Saint-Christophe-et-Nieves', 'Sainte-Lucie',
  'Saint-Vincent-et-les Grenadines', 'Salvador', 'Suriname',
  'Trinite-et-Tobago', 'Uruguay', 'Venezuela'
));

--Semi jointure sur les commandes Eur. N.
CREATE TABLE Commandes_EU_N AS(
SELECT * FROM Ryori.Commandes@DBLCentrale WHERE CODE_CLIENT
IN (SELECT DISTINCT CODE_CLIENT FROM Clients_EU_N)
);
--SELECT * FROM Clients_EU_N;
-- On verifie que les commandes ne proviennent que de ces clients

--Semi jointure sur les commandes des autres regions
CREATE TABLE Commandes_Autres AS(
SELECT * FROM Ryori.Commandes@DBLCentrale WHERE CODE_CLIENT
IN (SELECT DISTINCT CODE_CLIENT FROM Clients_AUTRES)
);
--SELECT * FROM Clients_AUTRES;
--On verifie que les commandes ne proviennent que de ces clients

--Semi jointure sur les details des commandes Eur. N.
CREATE TABLE Details_Commandes_EU_N AS(
SELECT * FROM Ryori.Details_Commandes@DBLCentrale
WHERE NO_COMMANDE
IN (SELECT DISTINCT NO_COMMANDE FROM Commandes_EU_N
));
--SELECT * FROM Details_Commandes_EU_N;
--SELECT * FROM Commandes_EU_N;
--On verifie que les num de commandes ne proviennent que de ces commandes

CREATE TABLE Details_Commandes_Autres AS(
SELECT * FROM Ryori.Details_Commandes@DBLCentrale
WHERE NO_COMMANDE
IN (SELECT DISTINCT NO_COMMANDE FROM Commandes_Autres
));
--SELECT * FROM Details_Commandes_Autres;
--SELECT * FROM Commandes_Autres;
--On verifie que les num de commandes ne proviennent que de ces commandes
```

Contraintes d'intégrité Lors de la création des fragments, sont importés tous les checks de la base centralisée de façon automatique. Il faut cependant ajouter les clés primaires et les clés étrangères. Pour les clés primaires, on travaille de manière classique :

```
--PK Clients

ALTER TABLE CLIENTS_EU_N
ADD CONSTRAINT PK_Clients_EU_N PRIMARY KEY (CODE_CLIENT);

ALTER TABLE CLIENTS_AUTRES
ADD CONSTRAINT PK_Clients_Autres PRIMARY KEY (CODE_CLIENT);

--PK Commandes

ALTER TABLE COMMANDES_EU_N
ADD CONSTRAINT PK_Commandes_EU_N PRIMARY KEY (NO_COMMANDE);

ALTER TABLE COMMANDES_AUTRES
ADD CONSTRAINT PK_Commandes_Autres PRIMARY KEY (NO_COMMANDE);

--PK Details commandes

ALTER TABLE DETAILS_COMMANDES_EU_N
ADD CONSTRAINT PK_Details_Commandes_EU_N PRIMARY KEY (NO_COMMANDE, REF_PRODUIT);

ALTER TABLE DETAILS_COMMANDES_AUTRES
ADD CONSTRAINT PK_Details_Commandes_Autres PRIMARY KEY (NO_COMMANDE, REF_PRODUIT);

--PK Fournisseurs

ALTER TABLE FOURNISSEURS
ADD CONSTRAINT PK_Fournisseurs PRIMARY KEY (NO_FOURNISSEUR);

--PK Stocks

ALTER TABLE STOCK_EU_N
ADD CONSTRAINT PK_Stock_EU_N PRIMARY KEY (REF_PRODUIT, PAYS);

ALTER TABLE STOCK_ALLEMAGNE
ADD CONSTRAINT PK_Stock_Allemagne PRIMARY KEY (REF_PRODUIT, PAYS);

ALTER TABLE STOCK_AUTRES
ADD CONSTRAINT PK_Stock_Autres PRIMARY KEY (REF_PRODUIT, PAYS);
```

Pour les clés étrangères on distingue deux cas. Dans le premier cas, si on crée des clés étrangères au sein d'une base locale, et auquel cas l'on crée les clés avec la syntaxe traditionnelle des foreign key.

```
--Cle etrangere des commandes vers leurs clients respectifs
ALTER TABLE COMMANDES_EU_N
ADD CONSTRAINT FK_Client_Commandes_EU_N FOREIGN KEY (CODE_CLIENT) REFERENCES CLIENTS_EU_N(CODE_CLIENT);

--Cle etrangere des commandes vers leurs clients respectifs
ALTER TABLE COMMANDES_AUTRES
ADD CONSTRAINT FK_Client_Commandes_Autres FOREIGN KEY (CODE_CLIENT) REFERENCES CLIENTS_AUTRES(CODE_CLIE

--Cle etrangere des details des commandes vers leurs commandes respectives
ALTER TABLE DETAILS_COMMANDES_EU_N
ADD CONSTRAINT FK_Commande_Details_Commandes_EU_N FOREIGN KEY (NO_COMMANDE) REFERENCES COMMANDES_EU_N(N

--Cle etrangere des details des commandes vers leurs commandes respectives
ALTER TABLE DETAILS_COMMANDES_AUTRES
ADD CONSTRAINT FK_Commande_Details_Commandes_Autres FOREIGN KEY (NO_COMMANDE) REFERENCES COMMANDES_AUTR
```

Dans le second cas, on veut créer une clé étrangère sur un attribut d'une table distante, on va donc créer un trigger dans les deux bases concernées par la relation de clé étrangère.
Ainsi, si l'on essaie de supprimer un attribut d'un tuple utilisé comme clé étrangère dans une table distante, on l'empêchera par le biais d'un trigger sur la suppression. Pour l'autre base (celle contenant la clé étrangère) on ajoutera un trigger à l'insertion ou à la mise à jour pour vérifier que la valeur existe bien dans la table distante.

```
-------------------------------------------------------
--  Trigger sur la suppression d'un fournisseur
-------------------------------------------------------

--Si le fournisseur est utilise dans une table distante, on empeche sa suppression

--Du cote de la BDD Europe du Sud on s'attend donc a trouver le second trigger qui
--verifiera en insertion/mise a jour que le Fournisseur existe bien sur notre BDD

CREATE OR REPLACE TRIGGER TRIGGER_DELETE_FOURNISSEURS
BEFORE DELETE ON FOURNISSEURS
FOR EACH ROW
DECLARE

  cpt integer;

BEGIN

  SELECT DISTINCT COUNT(*) INTO cpt
  FROM aouldhamou.produits@DBEuropeSud
  WHERE NO_FOURNISSEUR = :NEW.NO_FOURNISSEUR;

  IF (cpt <> 0) THEN
      RAISE_APPLICATION_ERROR(-20004, 'Fournisseur utilise, impossible de le supprimer');
  END IF;

END;
/

-------------------------------------------------------
--  Trigger sur l'insertion/mise a jour d'une commande
-------------------------------------------------------

--On verifie que l'employe realisant la commande existe
--bien dans la table distante (située en Amérique) des employes
```

```
--Du cote de la BDD Amerique on s'attend donc a trouver le second trigger qui
--verifiera en suppression que personne n'utilise l'employe a supprimer

CREATE OR REPLACE TRIGGER TRIGGER_FK_COMMANDES_AUTRES
BEFORE INSERT OR UPDATE ON COMMANDES_AUTRES
FOR EACH ROW

DECLARE

  cpt integer;

BEGIN

    SELECT DISTINCT COUNT(*)
        INTO cpt FROM tberthier.employes_am@DBAmerique
        WHERE NO_EMPLOYE = :NEW.NO_EMPLOYE;

    IF (cpt = 0) THEN
      RAISE_APPLICATION_ERROR(-20003, 'Employe inexistant');
    END IF;

END;
/

CREATE OR REPLACE TRIGGER TRIGGER_FK_COMMANDES_EU_N
BEFORE INSERT OR UPDATE ON COMMANDES_EU_N
FOR EACH ROW

DECLARE

  cpt integer;

BEGIN

    SELECT DISTINCT COUNT(*)
        INTO cpt FROM tberthier.employes_am@DBAmerique
        WHERE NO_EMPLOYE = :NEW.NO_EMPLOYE;

    IF (cpt = 0) THEN
      RAISE_APPLICATION_ERROR(-20003, 'Employe inexistant');
    END IF;

END;
/

-------------------------------------------------------
--  Trigger sur l'insertion/mise a jour des details d'une commande
-------------------------------------------------------

--On verifie que le produit existe bien dans la table distante
--(située en Eu.S.) des produits

--Du cote de la BDD Europe du Sud on s'attend donc a trouver le second trigger qui
--verifiera en suppression que personne n'utilise le produit a supprimer

CREATE OR REPLACE TRIGGER TRIGGER_FK_DETAILS_COM_AUTRES
BEFORE INSERT OR UPDATE ON DETAILS_COMMANDES_AUTRES
FOR EACH ROW

DECLARE
```

```
  cpt integer;

BEGIN

    SELECT DISTINCT COUNT(*)
        INTO cpt FROM aouldhamou.produits@DBEuropeSud
        WHERE REF_PRODUIT = :NEW.REF_PRODUIT;

    IF (cpt = 0) THEN
      RAISE_APPLICATION_ERROR(-20002, 'Produit inexistant');
    END IF;

END;
/

CREATE OR REPLACE TRIGGER TRIGGER_FK_DETAILS_COM_EU_N
BEFORE INSERT OR UPDATE ON DETAILS_COMMANDES_EU_N
FOR EACH ROW

DECLARE

  cpt integer;

BEGIN

    SELECT DISTINCT COUNT(*)
        INTO cpt FROM aouldhamou.produits@DBEuropeSud
        WHERE REF_PRODUIT = :NEW.REF_PRODUIT;

    IF (cpt = 0) THEN
      RAISE_APPLICATION_ERROR(-20002, 'Produit inexistant');
    END IF;

END;
/

------------------------------------------------------------
--  Trigger sur l'insertion/mise a jour du stock
------------------------------------------------------------

--On verifie que le produit existe bien dans la table distante
--(située en Eu.S.) des produits

--Du cote de la BDD Europe du Sud on s'attend donc a trouver le second trigger qui
--verifiera en suppression que personne n'utilise le produit a supprimer

CREATE OR REPLACE TRIGGER TRIGGER_FK_STOCK_ALLEMAGNE
BEFORE INSERT OR UPDATE ON STOCK_ALLEMAGNE
FOR EACH ROW

DECLARE

  cpt integer;

BEGIN

    SELECT DISTINCT COUNT(*)
        INTO cpt FROM aouldhamou.produits@DBEuropeSud
        WHERE REF_PRODUIT = :NEW.REF_PRODUIT;

    IF (cpt = 0) THEN
```

```
      RAISE_APPLICATION_ERROR(-20002, 'Produit inexistant');
    END IF;

END;
/

CREATE OR REPLACE TRIGGER TRIGGER_FK_STOCK_AUTRES
BEFORE INSERT OR UPDATE ON STOCK_AUTRES
FOR EACH ROW

DECLARE

  cpt integer;

BEGIN

    SELECT DISTINCT COUNT(*)
        INTO cpt FROM aouldhamou.produits@DBEuropeSud
        WHERE REF_PRODUIT = :NEW.REF_PRODUIT;

    IF (cpt = 0) THEN
      RAISE_APPLICATION_ERROR(-20002, 'Produit inexistant');
    END IF;

END;
/

CREATE OR REPLACE TRIGGER TRIGGER_FK_STOCK_EU_N
BEFORE INSERT OR UPDATE ON STOCK_EU_N
FOR EACH ROW

DECLARE

  cpt integer;

BEGIN

    SELECT DISTINCT COUNT(*) INTO cpt
        FROM aouldhamou.produits@DBEuropeSud
        WHERE REF_PRODUIT = :NEW.REF_PRODUIT;

    IF (cpt = 0) THEN
      RAISE_APPLICATION_ERROR(-20002, 'Produit inexistant');
    END IF;

END;
/
```

Il faut également rajouter de nouvelles contraintes pour s'assurer qu'on ne puisse pas rajouter des données incohérentes avec notre stratégie de fragmentation (empêcher de rajouter un client américain en Europe, par exemple).

```
ALTER TABLE CLIENTS_EU_N
ADD CONSTRAINT CHECK_Pays_Clients_EU_N CHECK (Pays IN (
  --Europe du Nord
  'Norvege', 'Suede', 'Danemark', 'Islande', 'Finlande', 'Royaume-Uni', 'Irlande', 'Belgique',
  'Luxembourg', 'Pays-Bas', 'Pologne', 'Allemagne'
));

ALTER TABLE CLIENTS_AUTRES
ADD CONSTRAINT CHECK_Pays_Clients_Autres CHECK (Pays NOT IN (
```

```
  --Europe du Nord
  'Norvege', 'Suede', 'Danemark', 'Islande', 'Finlande', 'Royaume-Uni', 'Irlande', 'Belgique',
  'Luxembourg', 'Pays-Bas', 'Pologne', 'Allemagne',
  --Europe du Sud
  'Espagne', 'Portugal', 'Andorre', 'France', 'Gibraltar', 'Italie', 'Saint-Marin', 'Vatican',
  'Malte', 'Albanie', 'Bosnie-Herzegovine', 'Croatie', 'Grece', 'Macedoine',
  'Montenegro', 'Serbie', 'Slovenie', 'Bulgarie',
  --Amerique
  'Antigua-et-Barbuda', 'Argentine', 'Bahamas', 'Barbade', 'Belize', 'Bolivie', 'Bresil',
  'Canada', 'Chili', 'Colombie', 'Costa Rica', 'Cuba', 'Republique dominicaine', 'Dominique',
  'Equateur', 'Etats-Unis', 'Grenade', 'Guatemala', 'Guyana', 'Haiti', 'Honduras', 'Jamaique',
  'Mexique', 'Nicaragua', 'Panama', 'Paraguay', 'Perou', 'Saint-Christophe-et-Nieves',
  'Sainte-Lucie', 'Saint-Vincent-et-les Grenadines', 'Salvador', 'Suriname',
  'Trinite-et-Tobago', 'Uruguay', 'Venezuela'
));

ALTER TABLE STOCK_EU_N
ADD CONSTRAINT CHECK_Pays_Stock_EU_N CHECK (Pays IN (
  --Europe du Nord hors Allemagne
  'Norvege', 'Suede', 'Danemark', 'Islande', 'Finlande', 'Royaume-Uni', 'Irlande', 'Belgique',
  'Luxembourg', 'Pays-Bas', 'Pologne'
));

ALTER TABLE STOCK_ALLEMAGNE
ADD CONSTRAINT CHECK_Pays_Stock_Allemagne CHECK (Pays = 'Allemagne');

ALTER TABLE STOCK_AUTRES
ADD CONSTRAINT CHECK_Pays_Stock_Autres CHECK (Pays NOT IN (
  --Europe du Nord
  'Norvege', 'Suede', 'Danemark', 'Islande', 'Finlande', 'Royaume-Uni', 'Irlande', 'Belgique',
  'Luxembourg', 'Pays-Bas', 'Pologne', 'Allemagne',
  --Europe du Sud
  'Espagne', 'Portugal', 'Andorre', 'France', 'Gibraltar', 'Italie', 'Saint-Marin', 'Vatican',
  'Malte', 'Albanie', 'Bosnie-Herzegovine', 'Croatie', 'Grece', 'Macedoine',
  'Montenegro', 'Serbie', 'Slovenie', 'Bulgarie',
  --Amerique
  'Antigua-et-Barbuda', 'Argentine', 'Bahamas', 'Barbade', 'Belize', 'Bolivie', 'Bresil',
  'Canada', 'Chili', 'Colombie', 'Costa Rica', 'Cuba', 'Republique dominicaine', 'Dominique',
  'Equateur', 'Etats-Unis', 'Grenade', 'Guatemala', 'Guyana', 'Haiti', 'Honduras', 'Jamaique',
  'Mexique', 'Nicaragua', 'Panama', 'Paraguay', 'Perou', 'Saint-Christophe-et-Nieves',
  'Sainte-Lucie', 'Saint-Vincent-et-les Grenadines', 'Salvador', 'Suriname',
  'Trinite-et-Tobago', 'Uruguay', 'Venezuela'
));
```

Droits d'accès Pour permettre aux autres bases de données de pouvoir lire le contenu de nos tables, on a besoin de leur donner les droits de sélection. On utilise un script pour nous générer toutes les requêtes : il envoie le résultat dans le fichier `droits_acces.sql` que l'on peut exécuter directement [1].

```
SET HEADING OFF;
SET PAGESIZE 0;
SET FEEDBACK OFF;
SET ECHO OFF;
SET VERIFY OFF;

SPOOL droits_acces.sql

SELECT 'GRANT SELECT ON ' || table_name || ' TO aouldhamou, tberthier;'
FROM user_tables;

SPOOL OFF;

@droits_acces.sql
```

1. Ceci permet d'éviter de recopier les affectations de droits pour chaque table.

Création des vues Les vues permettent l'interrogation de la base de données locale, comme si il s'agissait de la base de données centralisée. Elles utilisent donc le résultat de plusieurs sélections vers les bases de données Amérique et Europe de Sud pour recomposer les tables complètes.

```
--On verifie qu'on a les memes resultats avec Ryori.clients@DBCentrale

--Clients
CREATE VIEW CLIENTS AS
SELECT * FROM CLIENTS_EU_N
UNION
SELECT * FROM CLIENTS_AUTRES
UNION
SELECT * FROM aouldhamou.clients_eu_s@DBEuropeSud
UNION
SELECT * FROM tberthier.clients_am@DBAmerique;

--Commandes
CREATE VIEW COMMANDES AS
SELECT * FROM COMMANDES_EU_N
UNION
SELECT * FROM COMMANDES_AUTRES
UNION
SELECT * FROM aouldhamou.commandes_eu_s@DBEuropeSud
UNION
SELECT * FROM tberthier.commandes_am@DBAmerique;

--Details commandes
CREATE VIEW DETAILS_COMMANDES AS
SELECT * FROM DETAILS_COMMANDES_EU_N
UNION
SELECT * FROM DETAILS_COMMANDES_AUTRES
UNION
SELECT * FROM aouldhamou.details_commandes_eu_s@DBEuropeSud
UNION
SELECT * FROM tberthier.details_commandes_am@DBAmerique;

--Stock
CREATE VIEW STOCK AS
SELECT * FROM STOCK_EU_N
UNION
SELECT * FROM STOCK_ALLEMAGNE
UNION
SELECT * FROM STOCK_AUTRES
UNION
SELECT * FROM aouldhamou.stock_eu_s@DBEuropeSud
UNION
SELECT * FROM tberthier.stock_am@DBAmerique;

--Employes
CREATE VIEW EMPLOYES AS
SELECT * FROM tberthier.employes_am@DBAmerique;

--Produits
CREATE VIEW PRODUITS AS
SELECT * FROM aouldhamou.produits@DBEuropeSud;

--Categories
CREATE VIEW CATEGORIES AS
SELECT * FROM aouldhamou.categories@DBEuropeSud;
```

Nettoyages éventuels

```
-- Suppression des tables

DROP TABLE CLIENTS_EU_N CASCADE CONSTRAINTS;
DROP TABLE CLIENTS_AUTRES CASCADE CONSTRAINTS;
DROP TABLE COMMANDE_EU_N CASCADE CONSTRAINTS;
DROP TABLE CLIENTS_AUTRES CASCADE CONSTRAINTS;
DROP TABLE DETAILS_COMMANDES_EU_N CASCADE CONSTRAINTS;
DROP TABLE DETAILS_COMMANDES_AUTRES CASCADE CONSTRAINTS;
DROP TABLE FOURNISSEURS CASCADE CONSTRAINTS;
DROP TABLE STOCK_EU_N CASCADE CONSTRAINTS;
DROP TABLE STOCK_ALLEMAGNE CASCADE CONSTRAINTS;
DROP TABLE STOCK_AUTRES CASCADE CONSTRAINTS;

-- Suppression des vues

DROP VIEW CLIENTS;
DROP VIEW COMMANDES;
DROP VIEW DETAILS_COMMANDES;
DROP VIEW STOCK;
DROP VIEW EMPLOYES;
DROP VIEW PRODUITS;
DROP VIEW CATEGORIES;
```

Tests de vérification du bon fonctionnement

```
--Verifications, si la ligne est vide, c'est à dire qu'il n'y a
--aucune différence, les vues sont correctes
SELECT * FROM CLIENTS MINUS (SELECT * FROM Ryori.clients@DBLCentrale);
SELECT * FROM Ryori.clients@DBLCentrale MINUS (SELECT * FROM CLIENTS);

SELECT * FROM COMMANDES MINUS (SELECT * FROM Ryori.COMMANDES@DBLCentrale);
SELECT * FROM Ryori.COMMANDES@DBLCentrale MINUS (SELECT * FROM COMMANDES);

SELECT * FROM DETAILS_COMMANDES MINUS (SELECT * FROM Ryori.DETAILS_COMMANDES@DBLCentrale);
SELECT * FROM Ryori.DETAILS_COMMANDES@DBLCentrale MINUS (SELECT * FROM DETAILS_COMMANDES);

SELECT * FROM EMPLOYES MINUS (SELECT * FROM Ryori.EMPLOYES@DBLCentrale);
SELECT * FROM Ryori.clients@EMPLOYES MINUS (SELECT * FROM EMPLOYES);

SELECT * FROM STOCK MINUS (SELECT * FROM Ryori.STOCK@DBLCentrale);
SELECT * FROM Ryori.STOCK@DBLCentrale MINUS (SELECT * FROM STOCK);

SELECT * FROM PRODUITS MINUS (SELECT * FROM Ryori.PRODUITS@DBLCentrale);
SELECT * FROM Ryori.PRODUITS@DBLCentrale MINUS (SELECT * FROM PRODUITS);

SELECT * FROM CATEGORIES MINUS (SELECT * FROM Ryori.CATEGORIES@DBLCentrale);
SELECT * FROM Ryori.CATEGORIES@DBLCentrale MINUS (SELECT * FROM CATEGORIES);

SELECT * FROM FOURNISSEURS MINUS (SELECT * FROM Ryori.FOURNISSEURS@DBLCentrale);
SELECT * FROM Ryori.FOURNISSEURS@DBLCentrale MINUS (SELECT * FROM FOURNISSEURS);

--On obtient effectivement que des ensembles vides, donc on a bien
--la base centralisee equivalente a la base distribuee
```

III.C.2 Europe du Sud

Binôme responsable : B3111, composé de OULD HAMOUDA Aymen, LAFAILLE Yann.

Création des liens :

```
--Creation des liens
--BDCentrale :

    create DATABASE link linkToDBCentrale
    connect to aouldhamou
    identified by mdporacle781227
    using 'DB11';

--BDEuropeNord :

    create DATABASE link linkToDBEN
    connect to aouldhamou
    identified by MDPORACLE
    using 'DB12';

--BDAmerique :

    create DATABASE link linkToDBA
    connect to aouldhamou
    identified by MDPORACLE
    using 'DB14';
```

Création et peuplement des tables :

```
--Creation du fragment Europe du SUD

--creation les tables pour le fragment EU du sud

--Categories :
    create table Categories as (
    select *
    from Ryori.categories@linkToDBCentrale);
--Produits :
    create table produits as (
    select *
    from Ryori.produits@linkToDBCentrale);
--Stock :
   create table stock_eu_s as select *
    from Ryori.stock@linkToDBCentrale
        --permet de selectionner les tuples dont le pays est en Eu du sud
    where pays in('Espagne', 'Portugal', 'Andorre',
                'France', 'Gibraltar', 'Italie','Saint-Marin', 'Vatican',
                'Malte', 'Albanie', 'Bosnie-Herzegovine', 'Croatie',
                'Grece', 'Macedoine','Montenegro', 'Serbie', 'Slovenie', 'Bulgarie');



--Commandes :
 create table Commandes_eu_s as (
    select  Ryori.COMMANDES.NO_COMMANDE@linkToDBCentrale,
    Ryori.COMMANDES.CODE_CLIENT@linkToDBCentrale,
    Ryori.COMMANDES.NO_EMPLOYE@linkToDBCentrale,
    Ryori.COMMANDES.DATE_COMMANDE@linkToDBCentrale,
    Ryori.COMMANDES.DATE_ENVOI@linkToDBCentrale,
    Ryori.COMMANDES.PORT@linkToDBCentrale
    from Ryori.commandes@linkToDBCentrale,Clients_eu_s
    where Clients_eu_s.CODE_CLIENT = Ryori.commandes.CODE_CLIENT@linkToDBCentrale);

--Details_Commandes :
    create table details_commandes_eu_s as(
    select Ryori.details_commandes.no_commande@linkToDBCentrale,
    Ryori.details_commandes.ref_produit@linkToDBCentrale,
    Ryori.details_commandes.prix_unitaire@linkToDBCentrale,
    Ryori.details_commandes.quantite@linkToDBCentrale,
    Ryori.details_commandes.remise@linkToDBCentrale
    from Ryori.details_commandes@linkToDBCentrale,Produits,Commandes_EU_S
    where Produits.ref_produit=Ryori.details_commandes.ref_produit@linkToDBCentrale
    and Commandes_EU_S.no_commande=Ryori.details_commandes.no_commande@linkToDBCentrale);

--Clients :
    create table clients_eu_s as select *
    from Ryori.clients@linkToDBCentrale
    where pays in('Espagne', 'Portugal', 'Andorre', 'France', 'Gibraltar', 'Italie',
        'Saint-Marin', 'Vatican','Malte', 'Albanie', 'Bosnie-Herzegovine', 'Croatie', 'Grece',
        'Macedoine', 'Montenegro', 'Serbie', 'Slovenie', 'Bulgarie');
```

Clés primaires :

```
--Contraintes d'integrite pour le fragment Europe du Sud
--a-cles primaires

alter table categories ADD PRIMARY KEY (code_categorie);
alter table clients_eu_s ADD PRIMARY KEY (code_client);
alter table commandes_eu_s ADD PRIMARY KEY (no_commande);
alter table produits ADD PRIMARY KEY (ref_produit);
alter table stock_eu_s ADD PRIMARY KEY (pays, ref_produit);
alter table details_commandes_eu_s ADD PRIMARY KEY (no_commande,ref_produit);
```

Clés étrangères :

```
--Contraintes d'integrite
--b-clés étrangères

ALTER TABLE DETAILS_COMMANDES_EU_S
ADD CONSTRAINT FK_DETAILSCOMMANDES_PRODUITS
FOREIGN KEY (REF_PRODUIT) REFERENCES PRODUITS (REF_PRODUIT) ;

ALTER TABLE DETAILS_COMMANDES_EU_S
ADD CONSTRAINT FK_DETAILSCOMMANDES_COMMANDES
FOREIGN KEY (NO_COMMANDE) REFERENCES COMMANDES_EU_S (NO_COMMANDE) ;

ALTER TABLE STOCK_EU_S
ADD CONSTRAINT FK_STOCK_PRODUITS
FOREIGN KEY (REF_PRODUIT) REFERENCES PRODUITS (REF_PRODUIT) ;

ALTER TABLE COMMANDES_EU_S
ADD CONSTRAINT FK_COMMANDES_CLIENTS
FOREIGN KEY (CODE_CLIENT) REFERENCES CLIENTS_EU_S (CODE_CLIENT) ;

ALTER TABLE PRODUITS
ADD CONSTRAINT FK_PRODUITS_CATEGORIES
FOREIGN KEY (CODE_CATEGORIE) REFERENCES CATEGORIES (CODE_CATEGORIE) ;
```

Check :

```
--check
--permet de s'assurer que seulement des tuples concernant les pays
--d'europe du sud sont ajoutés aux tables client_eu_s
-- et stocks_eu_s

alter table clients_eu_s
add constraint check_pays_clients_eu_s
check (pays  in('Espagne', 'Portugal', 'Andorre', 'France',
                                'Gibraltar', 'Italie', 'Saint-Marin', 'Vatican',
                                'Malte', 'Albanie', 'Bosnie-Herzegovine', 'Croatie', 'Grece',
                                'Macedoine','Montenegro', 'Serbie', 'Slovenie', 'Bulgarie'));

alter table stock_eu_s
add constraint check_pays_stock_eu_s
check (pays  in('Espagne', 'Portugal', 'Andorre', 'France',
                                'Gibraltar', 'Italie', 'Saint-Marin', 'Vatican',
                                'Malte', 'Albanie', 'Bosnie-Herzegovine', 'Croatie', 'Grece',
                                'Macedoine','Montenegro', 'Serbie', 'Slovenie', 'Bulgarie'));
```

Triggers :

```
--d-Triggers
--trigger qui permet de ne pas affecter un numéro
--d'employe qui n'existe pas à une commande

create TRIGGER  fk_employe Before
INSERT ON commandes_eu_s
for each row
declare
cpt integer;
begin
  select distinct count(*) INTO cpt from  tberthier.employes_am@linkToDBA
  WHERE no_employe= :new.no_employe;
  if(cpt=0) then
  raise_application_error(-20002,'employé inexistant');
  end if;
end;

--trigger qui permet de ne pas affecter un numéro
--de fournisseurs qui n'existe pas à un produit

create TRIGGER  fk_no_fournisseurs Before
INSERT ON produits
for each row
declare
cpt integer;
begin
  select distinct count(*) INTO cpt from cjaverliat.fournisseurs@linkToDBEN
  WHERE no_fournisseur= :new.no_fournisseur;
  if(cpt=0) then
  raise_application_error(-20002,'fournisserus inexistant');
  end if;
end;
```

Droits d'accès :

```
--Droit sur select les tables du fragment EU du sud pour:
--Europe du Nord

grant select on clients_eu_s to cjaverliat;
grant select on Details_commandes_eu_s to cjaverliat;
grant select on produits to cjaverliat;
grant select on stock_eu_s to cjaverliat;
grant select on commandes_eu_s to cjaverliat;
grant select on categories to cjaverliat;

--Droits sur le selct sur les tables du fragment EU du sud pour:
--Amérique

grant select on clients_eu_s to tberthier;
grant select on Details_commandes_eu_s to tberthier;
grant select on produits to tberthier;
grant select on stock_eu_s to tberthier;
grant select on commandes_eu_s to tberthier;
grant select on categories to tberthier;
```

Création des vues et synonymes : cela permet l'interrogation de la base comme si elle était opérante en centralisée.

```
--Creation des vues du fragments EU du sud
--Clients
CREATE VIEW CLIENTS AS (
SELECT * FROM clients_eu_s
UNION ALL
SELECT * FROM cjaverliat.CLIENTS_EU_N@linkToDBEN
UNION ALL
SELECT * FROM cjaverliat.CLIENTS_AUTRES@linkToDBEN
UNION ALL
SELECT * FROM tberthier.CLIENTS_AM@linkToDBA
);
--Commandes

CREATE VIEW COMMANDES AS
SELECT * FROM COMMANDES_EU_S
UNION
SELECT * FROM cjaverliat.commandes_eu_n@linkToDBEN
UNION
SELECT * FROM cjaverliat.commandes_autres@linkToDBEN
UNION
SELECT * FROM tberthier.commandes_am@linktodba;



--Détails commandes

CREATE VIEW DETAILS_COMMANDES AS
SELECT REF_PRODUIT, NO_COMMANDE, PRIX_UNITAIRE, QUANTITE, REMISE
FROM cjaverliat.details_commandes_eu_n@linkToDBEN
UNION
SELECT REF_PRODUIT, NO_COMMANDE, PRIX_UNITAIRE, QUANTITE, REMISE
FROM cjaverliat.details_commandes_autres@linkToDBEN
UNION
SELECT REF_PRODUIT, NO_COMMANDE, PRIX_UNITAIRE, QUANTITE, REMISE
FROM details_commandes_eu_s
UNION
SELECT REF_PRODUIT, NO_COMMANDE, PRIX_UNITAIRE, QUANTITE, REMISE
FROM tberthier.details_commandes_am@linktodba;

--Stock

CREATE VIEW STOCK AS
SELECT * FROM STOCK_EU_S
UNION
SELECT * FROM cjaverliat.STOCK_EU_N@linkToDBEN
UNION
SELECT * FROM cjaverliat.STOCK_ALLEMAGNE@linkToDBEN
UNION
SELECT * FROM cjaverliat.STOCK_AUTRES@linkToDBEN
UNION
SELECT * FROM tberthier.stock_am@linktodba;

--Employes

CREATE VIEW EMPLOYES AS
SELECT * FROM tberthier.employes_am@linktodba;

--fournisseurs
```

```
Create view FOURNISSEURS AS
SELECT * FROM cjaverliat.fournisseurs@linkToDBEN
```

Code pour nettoyer le fragment :

```
--Netoyage du fragment EU du sud
-- En cas d'erreur
DROP TABLE Commandes_eu_s CASCADE CONSTRAINTS;
DROP TABLE Details_Commandes_eu_s CASCADE CONSTRAINTS;
DROP TABLE Stock_eu_s CASCADE CONSTRAINTS;
DROP TABLE Clients_eu_s CASCADE CONSTRAINTS;
DROP TABLE Categories CASCADE CONSTRAINTS;
DROP TABLE Produits CASCADE CONSTRAINTS;

-- Avant de commencer la replication
DROP VIEW CLIENTS;
DROP VIEW COMMANDES;
DROP VIEW DETAILS_COMMANDES;
DROP VIEW EMPLOYES;
DROP VIEW FOURNISSEURS;
DROP VIEW STOCK;
```

Vérification du bon fonctionnement de la table : on remarque que l'on obtient le même nombre de tuples selon la requête.

```
--On execute les commande suivantes et on voit bien qu'on obtient le meme nombre
--de tuples pour les requêtes sur les meme --tables.

select * from clients;
select * from Ryori.clients@linktodbcentrale;

select * from stock;
select * from Ryori.stock@linktodbcentrale;

select * from employes;
select * from Ryori.employes@linktodbcentrale;

select * from fournisseurs;
select * from Ryori.fournisseurs@linktodbcentrale;

select * from commandes;
select * from Ryori.commandes@linktodbcentrale;

select * from details_commandes;
select * from Ryori.details_commandes@linktodbcentrale;

select * from produits;
select * from Ryori.produits@linktodbcentrale;

select * from categories;
select * from Ryori.categories@linktodbcentrale;
```

III.C.3 Amérique

On procède de la même façon pour le site Amérique :

```
--avoir acces a DB11 en etant connecte sur DB14
create database link DBL_principale
connect to pgevraud
identified by mdporacle
using 'DB11';

drop database link dba_eu_nord;

--donner acces aux tables qu'on crees
grant select, update, insert, delete
on X to pgevraud;
-- avoir acces aux tables creees par tberthier sur DB14
CREATE SYNONYM X FOR tberthier.X;
--supprimer le synonym
drop synonym X;

--creation des tables
create table Stock_AM as
(select * from Ryori.stock@DBL_principale
where pays in (--Amerique
  'Antigua-et-Barbuda', 'Argentine', 'Bahamas', 'Barbade', 'Belize', 'Bolivie',
  'Bresil', 'Canada', 'Chili', 'Colombie', 'Costa Rica', 'Cuba', 'Republique dominicaine',
  'Dominique', 'Equateur', 'Etats-Unis', 'Grenade', 'Guatemala', 'Guyana', 'Haiti',
  'Honduras', 'Jamaique', 'Mexique', 'Nicaragua', 'Panama', 'Paraguay', 'Perou',
  'Saint-Christophe-et-Nieves', 'Sainte-Lucie', 'Saint-Vincent-et-les Grenadines',
  'Salvador', 'Suriname', 'Trinite-et-Tobago', 'Uruguay', 'Venezuela');


CREATE TABLE Commandes_AM AS(
SELECT * FROM Ryori.Commandes@DBL_principale WHERE CODE_CLIENT IN (SELECT DISTINCT CODE_CLIENT FROM Cli
);

CREATE TABLE Details_Commandes_AM AS(
SELECT * FROM Ryori.Details_Commandes@DBL_principale WHERE NO_COMMANDE IN
(SELECT DISTINCT NO_COMMANDE FROM Commandes_AM
));

--foreign key pour employes
alter table employes_am
add constraint FKEYemployes foreign key (REND_COMPTE) references employes_am(NO_EMPLOYE);

--trigger qui modelise la foreign key de details_commandes
CREATE OR REPLACE TRIGGER TRIGGER_FK_DETAILS_COMMANDES
BEFORE INSERT OR UPDATE ON DETAILS_COMMANDES_AM
FOR EACH ROW
DECLARE
   cpt integer;
BEGIN
        select count(*) into cpt from produits where :NEW.REF_PRODUIT=produits.REF_PRODUIT;
        IF(cpt=0) then
                RAISE_APPLICATION_ERROR(-20001,'Aucun produit correspondant a cette reference');
  END IF;
END;


--trigger qui modelise la foreign key de stock
CREATE OR REPLACE TRIGGER TRIGGER_FK_STOCK
BEFORE INSERT OR UPDATE ON STOCK_AM
```

```
FOR EACH ROW
DECLARE
   cpt integer;
BEGIN
        select count(*) into cpt from produits where :NEW.REF_PRODUIT=produits.REF_PRODUIT;
        IF(cpt=0) then
                RAISE_APPLICATION_ERROR(-20001,'Aucun produit correspondant a cette reference');
  END IF;
END;
```

```
CREATE OR REPLACE TRIGGER TRIGGER_FK_EMPLOYES
BEFORE delete ON employes_am
FOR EACH ROW
DECLARE
   cpt integer;
BEGIN
        select count(*) into cpt from (commandes_eur_n union commandes_eur_s)
        where (:NEW.no_employe=commandes_eur_s.no_employe
                or :NEW.no_employe=commandes_eur_n.no_employe);
        IF(cpt<>0) then
                RAISE_APPLICATION_ERROR(-20003,'Cet employe est enrole dans une commande');
  END IF;
END;
```

Pour les vues et l'utilisation de synonymes, on a :

```
--Clients
CREATE VIEW CLIENTS AS
SELECT * FROM cjaverliat.CLIENTS_EU_N@dbl_eu_n
UNION
SELECT * FROM cjaverliat.CLIENTS_AUTRES@dbl_eu_n
UNION
SELECT * FROM aouldhamou.clients_eu_s@DBl_eu_s
UNION
SELECT * FROM clients_am;

--Commandes
CREATE VIEW COMMANDES AS
SELECT * FROM cjaverliat.COMMANDES_EU_N@dbl_eu_n
UNION
SELECT * FROM  cjaverliat.COMMANDES_AUTRES@dbl_eu_n
UNION
SELECT * FROM aouldhamou.commandes_eu_s@DBl_eu_s
UNION
SELECT * FROM commandes_am;

--Details commandes
CREATE VIEW DETAILS_COMMANDES AS

SELECT REF_PRODUIT, NO_COMMANDE, PRIX_UNITAIRE, QUANTITE, REMISE
FROM cjaverliat.DETAILS_COMMANDES_EU_N@dbl_eu_n
UNION
SELECT REF_PRODUIT, NO_COMMANDE, PRIX_UNITAIRE, QUANTITE, REMISE
FROM  cjaverliat.DETAILS_COMMANDES_AUTRES@dbl_eu_n
UNION
SELECT REF_PRODUIT, NO_COMMANDE, PRIX_UNITAIRE, QUANTITE, REMISE
FROM aouldhamou.details_commandes_eu_s@DBl_eu_s
UNION
SELECT REF_PRODUIT, NO_COMMANDE, PRIX_UNITAIRE, QUANTITE, REMISE
FROM details_commandes_am;

-- Test
SELECT * FROM cjaverliat.DETAILS_COMMANDES_EU_n@dbl_eu_n;


--Stock
CREATE VIEW STOCK AS
SELECT * FROM cjaverliat.STOCK_EU_N@dbl_eu_n
UNION
SELECT * FROM  cjaverliat.STOCK_ALLEMAGNE@dbl_eu_n
UNION
SELECT * FROM  cjaverliat.STOCK_AUTRES@dbl_eu_n
UNION
SELECT * FROM aouldhamou.stock_eu_s@dbl_eu_s
UNION
SELECT * FROM stock_am;


--Produits
CREATE VIEW PROD AS
SELECT * from produits; --ici produits est un synonyme
```

IV Tests de requête distribuées et optimisations

IV.A Europe du Sud

Binôme responsable : B3111, OULD HAMOUDA Aymen, LAFAILLE Yann

IV.A.1 Première requête

```
-- Requete 1 test BD distribuée
select * from Clients;
```

```
BOTTM Bottom-Dollar Markets              23 Tsawassen Blvd.                                        Tsawassen
CACTU Cactus Comidas para llevar         Cerrito 333                                               Buenos Aires
CENTC Centro comercial Moctezuma         Sierras de Granada 9993                                   Mexico D.F.
COMMI ComÈrcio Mineiro                   Av. dos LusÌadas, 23                                      S„o Paulo

CODE_ SOCIETE                            ADRESSE                                                   VILLE
----- ---------------------------------- --------------------------------------------------------- ---------------------------
FAMIA Familia Arquibaldo                 Rua Orus, 92                                              S„o Paulo
GOURL Gourmet Lanchonetes                Av. Brasil, 442                                           Campinas
GREAL Great Lakes Food Market            2732 Baker Blvd.                                          Eugene
GROSR GROSELLA-Restaurante               5ter Ave. Los Palos Grandes                               Caracas
HANAR Hanari Carnes                      Rua do PaÁo, 67                                           Rio de Janeiro
HILAA HILARI"N-Abastos                   Carrera 22 con Ave. Carlos Soublette #8-35                San CristÛbal
HUNGC Hungry Coyote Import Store         City Center Plaza516 Main St.                             Elgin
LAUGB Laughing Bacchus Wine Cellars      1900 Oak St.                                              Vancouver
LAZYK Lazy K Kountry Store               12 Orchestra Terrace                                      Walla Walla
LETSS Let's Stop N Shop                  87 Polk St.Suite 5                                        San Francisco
LILAS LILA-Supermercado                  Carrera 52 con Ave. Bolivar #65-98 Liano Largo            Barquisimeto

CODE_ SOCIETE                            ADRESSE                                                   VILLE
----- ---------------------------------- --------------------------------------------------------- ---------------------------
LINOD LINO-Delicateses                   Ave. 5 de Mayo Porlamar                                   I. de Margarita
LONEP Lonesome Pine Restaurant           89 Chiaroscuro Rd.                                        Portland
MEREP MËre Paillarde                     43 rue St. Laurent                                        Montreal

91 lignes sélectionnées.
```

FIGURE 1 – Première requête : résultat

Cette requête nous retourne le bon nombre de tuple, qui est le même résultat que la requête :

```
SELECT * FROM
Ryori.client@linkToDBCentrale
```

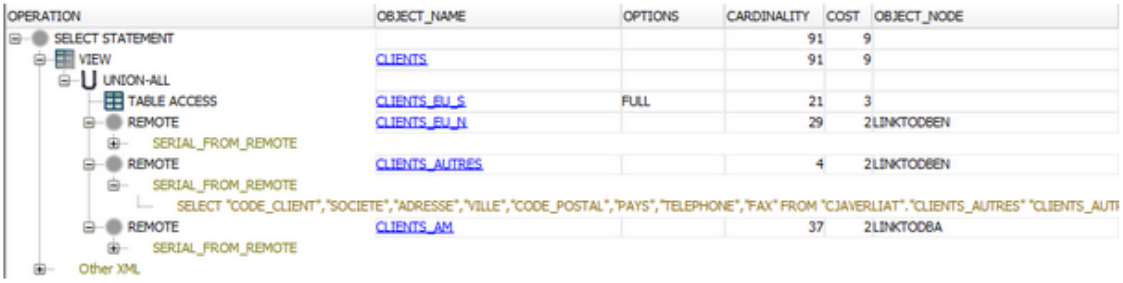


FIGURE 2 – Première requête : analyse

IV.A.2 Deuxième requête

```
--Requete 2 test BD distribuée
select * from clients where pays in('France','Espagne','Italie');
```

CODE_	SOCIETE	ADRESSE	VILLE
FOLIG	Folies gourmandes	184, chaussee de Tournai	Lille
FRANR	France restauration	54, rue Royale	Nantes
FRANS	Franchi S.p.A.	Via Monte Bianco 34	Torino
GALED	Galeria del gastrunomo	Rambla de Cataluoa, 23	Barcelona
GODOS	Godos Cocina TÌpica	Romero, 33	Sevilla
LACOR	La corne d'abondance	67, avenue de l'Europe	Versailles
LAMAI	La maison d'Asie	1 rue Alsace-Lorraine	Toulouse
MAGAA	Magazzini Alimentari Riuniti	Via Ludovico il Moro 22	Bergamo

19 lignes sélectionnées.

FIGURE 3 – Deuxième requête : résultat

Cette requête nous retourne le bon nombre de tuple (le même résultat que la requête)

```
SELECT *
FROM Ryori.client@linkToDBCentrale
WHERE pays in('France','Espagne','Italie');
```

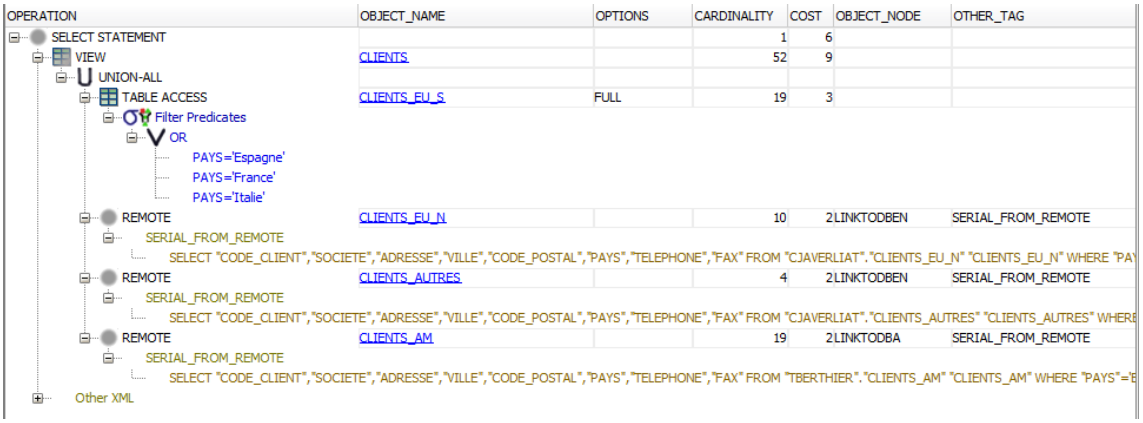


FIGURE 4 – Deuxième requête : analyse

IV.A.3 Troisième requête

```
--Requete 3 test BD distribuée
select * from clients where pays in('France','Espagne','Italie');
```

CODE_	SOCIETE	ADRESSE	VILLE	CODE_POSTA	PAYS	TELEPHONE
FOLIG	Folies gourmandes	184, chaussee de Tournai	Lille	59000	France	03.20.16.10.16
FRANR	France restauration	54, rue Royale	Nantes	44000	France	02.40.32.21.21
FRANS	Franchi S.p.A.	Via Monte Bianco 34	Torino	10100	Italie	011-4988260
GALED	Galeria del gastronomo	Rambla de Cataluoa, 23	Barcelona	08022	Espagne	(93) 203 4560
GODOS	Godos Cocina TÍpica	Romero, 33	Sevilla	41101	Espagne	(95) 555 82 82
LACOR	La corne d'abondance	67, avenue de l'Europe	Versailles	78000	France	01.30.59.84.10
LAMAI	La maison d'Asie	1 rue Alsace-Lorraine	Toulouse	31000	France	05.61.77.61.10
MAGAA	Magazzini Alimentari Riuniti	Via Ludovico il Moro 22	Bergamo	24100	Italie	035-640230

19 lignes sélectionnées.

FIGURE 5 – Troisième requête : résultat

Cette requête nous retourne le bon nombre de tuple (le même résultat que la requête 2 ci-dessus)

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			19	3
TABLE ACCESS	CLIENTS_EU_S	FULL	19	3
Filter Predicates				
OR				
PAYS='Espagne'				
PAYS='France'				
PAYS='Italie'				
Other XML				

FIGURE 6 – Troisième requête : analyse

On remarque ici que nous avons un coût de 3 pour cette requête, alors que nous avions un coût de 9 pour la requête 3. La raison est que ici on ne fait pas de remote acces.

IV.A.4 Quatrième requête

```
--Requete 4 test BD distribuée
select * from employes;
```

NO_EMPLOYE	REND_COMPTE	NOM	PRENOM	FONCTION	TITRE	DAT
2	2	Fuller	Andrew	Vice-PrÈsident	Dr.	19/
8	2	Callahan	Laura	Assistante commerciale	Mlle	09/
4	2	Peacock	Margaret	ReprÈsentant(e)	Mme	19/
3	2	Leverling	Janet	ReprÈsentant(e)	Mlle	30/
1	2	Davolio	Nancy	ReprÈsentant(e)	Mlle	08/
5	2	Buchanan	Steven	Chef des ventes	M.	04/
9	5	Dodsworth	Anne	ReprÈsentant(e)	Mlle	02/
6	5	Suyama	Michael	ReprÈsentant(e)	M.	02/
7		King	Robert	ReprÈsentant(e)	M.	29/

9 lignes sélectionnées.

FIGURE 7 – Quatrième requête : résultat

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST	OBJECT_NODE
SELECT STATEMENT		REMOTE	9	3	
TABLE ACCESS	TBERTHIER.EMPLOYES_AM	FULL	9	3	DB14

FIGURE 8 – Quatrième requête : analyse

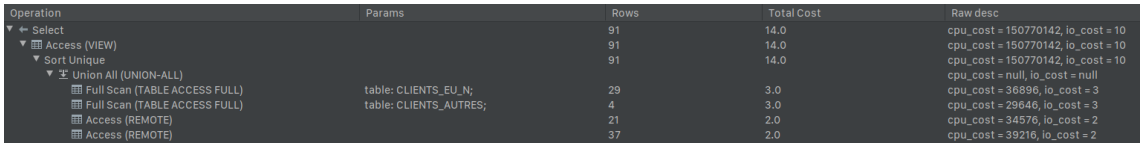
IV.B Europe du Nord

IV.B.1 Sélection de tous les clients

```
SELECT * FROM CLIENTS;
```

Résultat d'éxecution On obtient le même résultat que la sélection de tous les clients sur la base de données centralisée (91 lignes).

Analyse du plan d'exécution On constate qu'on fait 2 accès à des tables locales (CLIENTS_EU_N et CLIENTS_AUTRES) ainsi que 2 accès distants, il s'agit des accès réalisés par la vue Clients vers les tables des clients en Europe du Sud et en Amérique, on remarquera que le coût calculé est de 14.



Operation	Params	Rows	Total Cost	Raw desc
Select		91	14.0	cpu_cost = 150770142, io_cost = 10
Access (VIEW)		91	14.0	cpu_cost = 150770142, io_cost = 10
Sort Unique		91	14.0	cpu_cost = 150770142, io_cost = 10
Union All (UNION-ALL)				cpu_cost = null, io_cost = null
Full Scan (TABLE ACCESS FULL)	table: CLIENTS_EU_N;	29	3.0	cpu_cost = 36896, io_cost = 3
Full Scan (TABLE ACCESS FULL)	table: CLIENTS_AUTRES;	4	3.0	cpu_cost = 29646, io_cost = 3
Access (REMOTE)		21	2.0	cpu_cost = 34576, io_cost = 2
Access (REMOTE)		37	2.0	cpu_cost = 39216, io_cost = 2

FIGURE 9 – Plan d'exécution de la requête

Autre écriture possible Autrement dit, la même requête peut s'écrire :

```
SELECT * FROM CLIENTS_EU_N
UNION
SELECT * FROM CLIENTS_AUTRES
UNION
SELECT * FROM aouldhamou.clients_eu_s@DBEuropeSud
UNION
SELECT * FROM tberthier.clients_am@DBAmerique;
```

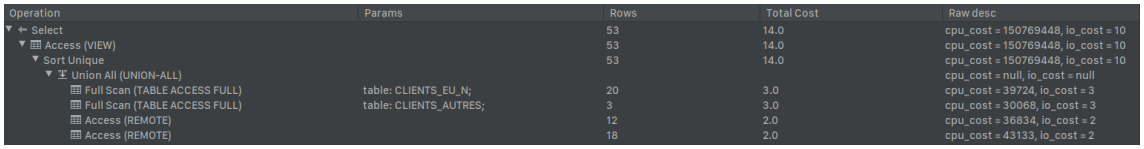
IV.B.2 Sélection d'un sous-ensemble de clients de l'Europe du Nord

Dans ce cas on essaie de récupérer les clients du Royaume-Uni, du Danemark, et de l'Allemagne.

```
SELECT * FROM CLIENTS WHERE Pays IN ('Royaume-Uni', 'Danemark', 'Allemagne');
```

Résultat d'éxecution On obtient un ensemble de 20 clients.

Analyse du plan d'exécution On constate qu'on fait 2 accès à des tables locales (CLIENTS_EU_N et CLIENTS_AUTRES) ainsi que 2 accès distants, comme pour la première requête. Néanmoins cette requête est criticable, car nos clients se trouvent tous dans CLIENTS_EU_N. Il y a donc 3 accès à des tables inutiles dans cette requête. Le coût peut être réduit.



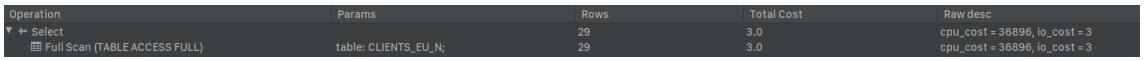
Operation	Params	Rows	Total Cost	Raw desc
Select		53	14.0	cpu_cost = 150769448, io_cost = 10
Access (VIEW)		53	14.0	cpu_cost = 150769448, io_cost = 10
Sort Unique		53	14.0	cpu_cost = 150769448, io_cost = 10
Union All (UNION-ALL)				cpu_cost = null, io_cost = null
Full Scan (TABLE ACCESS FULL)	table: CLIENTS_EU_N;	20	3.0	cpu_cost = 39724, io_cost = 3
Full Scan (TABLE ACCESS FULL)	table: CLIENTS_AUTRES;	3	3.0	cpu_cost = 30068, io_cost = 3
Access (REMOTE)		12	2.0	cpu_cost = 36834, io_cost = 2
Access (REMOTE)		18	2.0	cpu_cost = 43133, io_cost = 2

FIGURE 10 – Plan d'exécution de la requête

Autre écriture possible

```
SELECT * FROM CLIENTS_EU_N WHERE Pays IN ('Royaume-Uni', 'Danemark', 'Allemagne');
```

Avec cette requête on a un seul accès à la table CLIENTS_EU_N, ce qui est beaucoup plus optimisé. C'est tout l'intérêt de notre fragmentation.



Operation	Params	Rows	Total Cost	Raw desc
Select		29	3.0	cpu_cost = 36896, io_cost = 3
Full Scan (TABLE ACCESS FULL)	table: CLIENTS_EU_N;	29	3.0	cpu_cost = 36896, io_cost = 3

FIGURE 11 – Plan d'exécution de la requête

On portera une attention toute particulière au coût de la requête, ainsi passé de 14 à 3. Ce coût se traduit directement par une vitesse d'exécution de la requête plus rapide (100ms -> 50ms).

IV.C Amérique

Afin de tester si nos fragments ont bien été créés, on compare le contenu de nos tables avec le contenu des tables de Ryori, en respectant les conditions de la fragmentation bien évidemment. Pour ce faire, on utilise l'opérateur minus, qui renvoie les tuples existant dans première table, mais pas dans la seconde. Cependant, pour être sur qu'il n'existe pas des tuples dans la seconde qui n'existeraient pas dans la première, il faut tester avec l'opérateur minus "dans les deux sens" : A-B, puis B-A. Si notre fragmentation a bien été effectuée, on obtiendra un résultat nul pour chacun des tests.

Clients :

FIGURE 12 – Exécution de la requête pour clients

Employés :

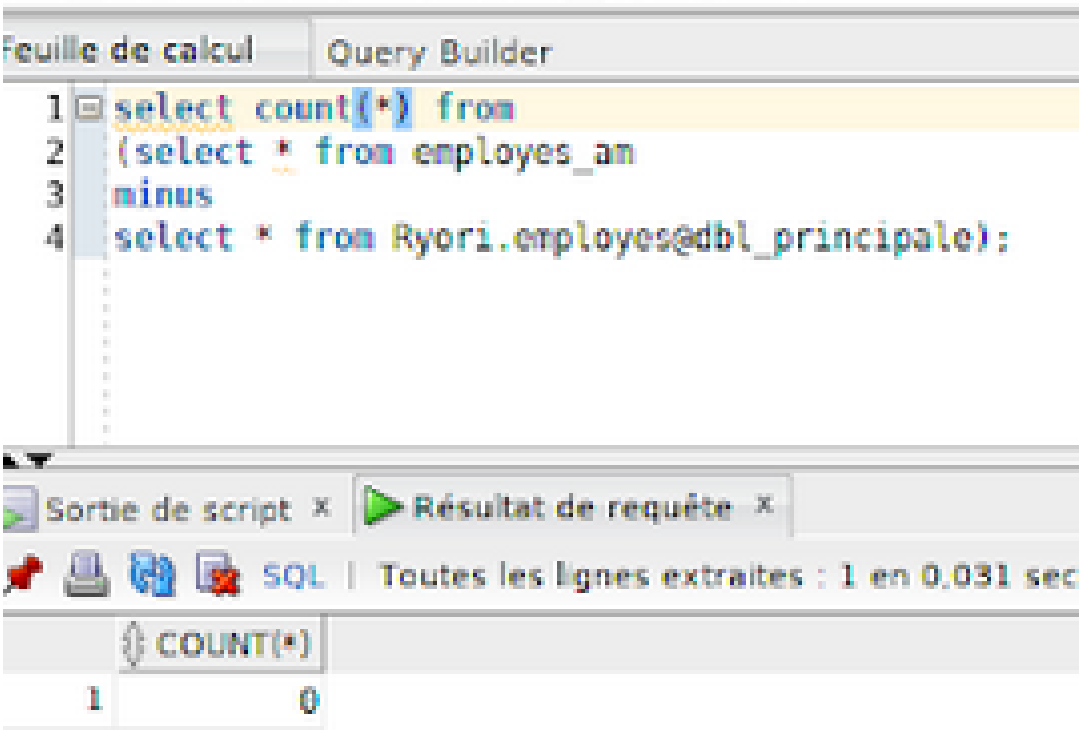


FIGURE 13 – Exécution de la requête pour employés

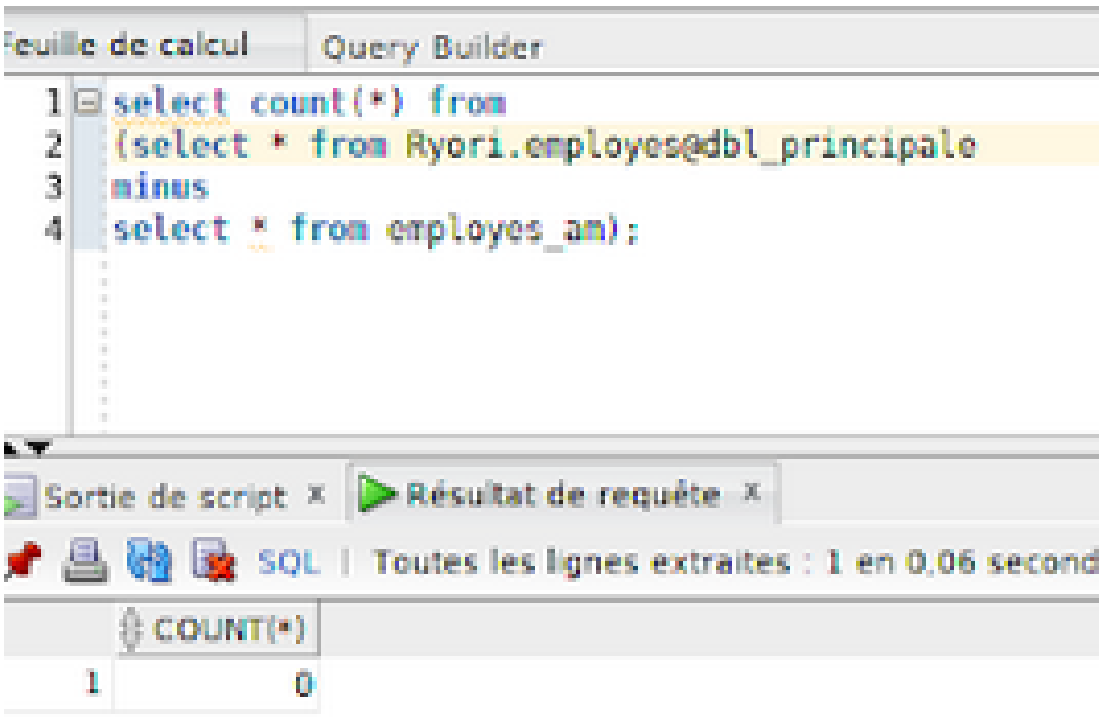


FIGURE 14 – Exécution de la requête pour employés (bis)

Stock :

FIGURE 15 – Exécution de la requête pour stock

FIGURE 16 – Exécution de la requête pour stock (bis)

Commandes :

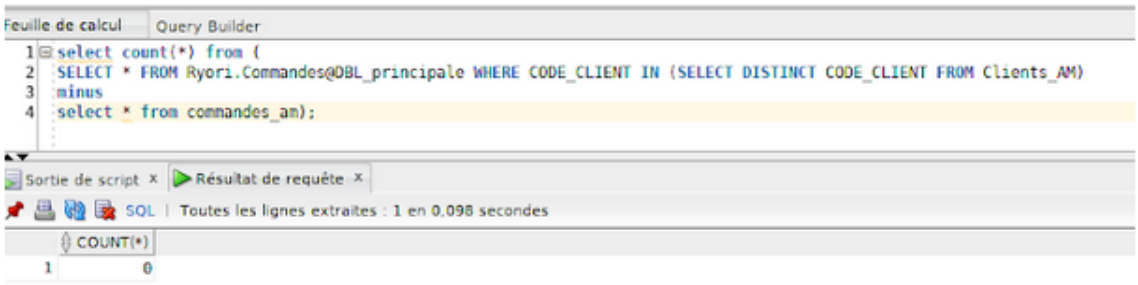


FIGURE 17 – Exécution de la requête pour commandes

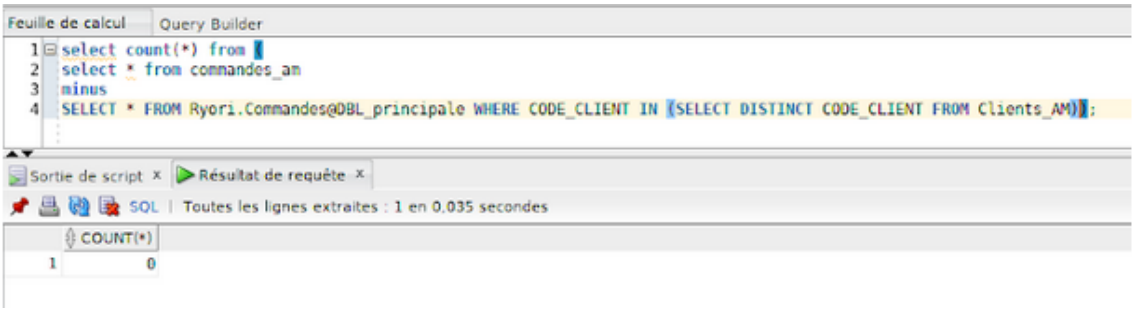


FIGURE 18 – Exécution de la requête pour commandes (bis)

Détails commande :

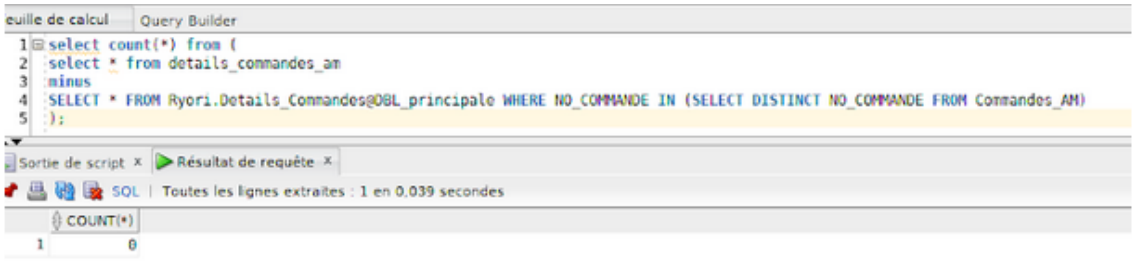


FIGURE 19 – Exécution de la requête pour détails commandes

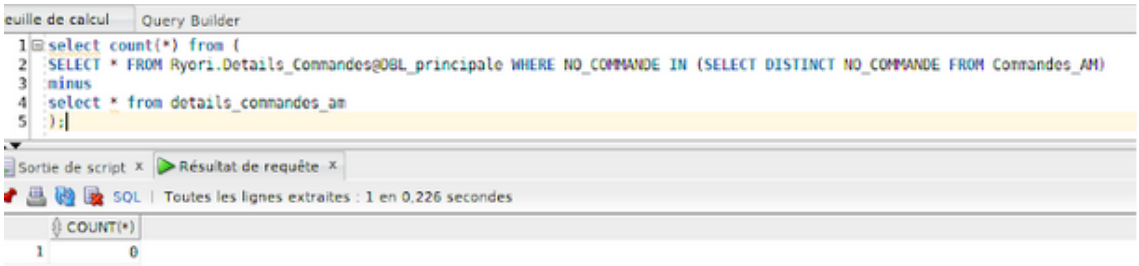


FIGURE 20 – Exécution de la requête pour détails commandes (bis)

V Réplications

V.A Europe du Nord

Rappel binôme responsable : Charles Javerliat, Pierre Sibut-Bourde `B3109`.

V.A.1 Objectifs

L'objectif de la réplication est de faciliter l'accès aux données car elles seront locales (par copie) au lieu d'être à distance.

V.A.2 Liste des réplications prévues

Trois réplications vont s'opérer sur ce site :
— Employés
— Produits
— Catégories

V.A.3 Analyse

Fait ci-dessous avec la réplication associée.

V.A.4 Mise en œuvre des réplications pour des besoins locaux

On utilise à ce sujet :

```
CREATE MATERIALIZED VIEW MVR_Employes
REFRESH FAST
NEXT sysdate + (1/24/60)
AS
SELECT *
FROM TBERTHIER.EMPLOYES_AM@DBAmerique;

CREATE MATERIALIZED VIEW MVR_Produits
REFRESH FAST
NEXT sysdate + (1/24/60)
AS
SELECT * FROM AOULDHAMOU.PRODUITS@DBEuropeSud;

CREATE MATERIALIZED VIEW MVR_Categories
REFRESH COMPLETE
NEXT sysdate + (1/24/60)
AS
SELECT * FROM AOULDHAMOU.CATEGORIES@DBEuropeSud;
```

Dans le cas des refresh-fast, on a besoin que le site maître (celui qui possède les données) crée un fichier de log pour que le refresh accède à l'historique des modifications/différences.

Produits Nous avons fait la réplication de la table `Produits` en refresh-fast car il est peu probable que les mises à jour faites sur cette table dépassent plus de 50% de sa cardinalité.

Employes Nous avons utilisé le refresh-fast, même si un complete aurait été également possible en supposant que les employés changent régulièrement en masse (cas des emplois intérimaires par exemple).

Categories Nous avons fait la réplication de la table `Produits` en refresh-complete pour essayer cette méthode en plus du fast, sans raison particulière.

Pour l'Europe du Sud :
— Message émis à l'Europe du Sud : créer un fichier de log pour Produits
— Réponse du site maître :

```
CREATE MATERIALIZED VIEW LOG ON PRODUITS;
GRANT SELECT ON MLOG$_PRODUITS TO cjaverliat;
```

— Tests de vérification de bon fonctionnement de la réplication : après que l'Europe du Sud ait rajouté un nouveau produit dans leur table, on le voit bien apparaitre dans notre réplicat.

Pour l'Amérique :

— Message émis à l'Amérique : créer un fichier de log pour Employes

— Réponse du site maître :

```
CREATE MATERIALIZED VIEW LOG ON EMPLOYES;
GRANT SELECT ON MLOG$_EMPLOYES TO cjaverliat;
```

— Tests de vérification de bon fonctionnement de la réplication : après que l'Amérique ait rajouté un nouvel employé dans leur table, on le voit bien apparaitre dans notre réplicat.

V.A.5 Mise à jour des vues

On met à jour nos vues pour utiliser les réplicats :

```
-----------------------------------------
-- On peut modifier les vues precedemment
-- creees pour utiliser les replicats
-----------------------------------------

--Employés
CREATE VIEW EMPLOYES AS
SELECT * FROM MVR_EMPLOYES;

--Produits
CREATE VIEW PRODUITS AS
SELECT * FROM MVR_PRODUITS;

--Catégories
CREATE VIEW CATEGORIES AS
SELECT * FROM MVR_CATEGORIES;
```

Dans tous les cas, on vérifie le bon fonctionnement de la réplication en utilisant `INSERT` et en vérifiant que le tuple inséré est apparent dans le réplicat des autres bases de données distantes en communiquant avec nos collègues.

V.A.6 Création de logs pour les autres sites

Nous avons reçu une demande de nos collègues américains de création de logs pour qu'ils puissent utiliser les refresh-fast sur notre table Fournisseurs :

— Message reçu par l'Amérique : créer un fichier de log pour Fournisseurs

— Réponse :

```
CREATE MATERIALIZED VIEW LOG ON FOURNISSEURS;
GRANT SELECT ON MLOG$_FOURNISSEURS TO aouldhamou, tberthier;
```

— Tests de vérification de bon fonctionnement de la réplication : après avoir rajouté un nouveau fournisseur dans notre table, les américains le voient bien apparaitre dans leur réplicat.

V.B Europe du Sud

Rappel binôme reponsable : OULD HAMOUDA Aymen, LAFAILLE Yann, B3111

V.B.1 Objectifs

L'objectif de la réplication est de faciliter l'accès aux données car elles seront locales (par copie) au lieu d'être à distance.

V.B.2 Liste des réplications prévues

Deux réplications sont prévues pour ce site :
— Employés
— Fournisseurs
Comme les tables Produits et Catégories sont locales, il n'y aura pas de réplication.

V.B.3 Analyse

Table EMPLOYES :
Nous avons fait la réplication de la table EMPLOYES_AM en fast car nous avons estimé qu'il y aurait peu de chances pour que les mises à jour faites sur cette table dépasse plus de 50% de sa cardinalité.

Table FOURNISSEURS :
Nous avons fait la réplication de la table FOURNISSEURS en complete car nous avons estimé que les mises à jour faites sur cette table pourraient dépasser plus de 50% de sa cardinalité.

V.B.4 Mise en œuvre des réplications pour les besoins locaux

Fournisseurs
— Opéations réalisées localement

```
-- Mise en place du réplicat du fragment Fournisseurs
-- Opérations réalisées localement

CREATE MATERIALIZED VIEW DMV_FOURNISSEURS
REFRESH COMPLETE
NEXT SYSDATE +(1/24/60)
AS
SELECT * FROM cjaverliat.fournisseurs@linktodben;
```

— Message émis au site Europe du Nord : créer des logs pour Fournisseurs
— Réponse du site maître :

```
CREATE MATERIALIZED VIEW LOG ON FOURNISSEURS;
GRANT SELECT ON MLOG$_FOURNISSEURS TO aouldhamou, tberthier;
```

— Tests de vérification de bon fonctionnement de la réplication : on fait un INSERT dans la table chez le site maître et l'on vérifie que ce qui est inséré apparaît bien dans la réplication réalisée.
On fait :

```
insert into fournisseurs
values (30,'Holmes & Co','221b Baker St','London','EC1',
'SD','Royaume-Uni','(420) 696-9696',null);
```

Et l'on vérifie alors bien que le tuple est inséré (c'est le cas ici).
— Évolutions éventuelles des contraintes d'intégrité : il faut modifier le déclencheur fk_no_fournisseur pour que ce dernier utilise la vue matérialisée et non l'accès à distance par lien.

Employés
— Opéations réalisées localement

```
--Mise en place du replicat du fragment Employes
--Opérations réalisées localement

CREATE MATERIALIZED VIEW DMV_EMPLOYES
REFRESH FAST
NEXT SYSDATE +(1/24/60)
```

```
AS
SELECT * FROM tberthier.employes_am@linktodba;
```

— Message émis au site Amérique : demande de *materialized view log* sur Employés sur le site Amérique
— Réponse du site maître :

```
--Reponse du site maitre

CREATE MATERIALIZED VIEW LOG ON EMPLOYES;
GRANT SELECT ON MLOG$_EMPLOYES TO AOULDHAMOU;
```

— Tests de vérification de bon fonctionnement de la réplication : on fait un `INSERT` dans la table chez le site maître et l'on vérifie que ce qui est inséré apparaît bien dans la réplication réalisée. On fait :

```
insert into employes values
(10,10,'a','b','c','d','10/10/10','10/10/10',10,10);
```

Puis l'on vérifie alors bien que le tuple est inséré (c'est le cas ici).
— Évolutions éventuelles des contraintes d'intégrité : il faut modifier le déclencheur pour que ce dernier utilie la vue matérialisée et non l'accès à distance par liens.
— Évolutions éventuelles des vues et des synonymes : il faut créer un synonyme pour la table. On fait `CREATE SYNONYM Employes FOR DMV_EMPLOYES;`.

V.C Amérique

On procède :

```
--Fournisseurs
CREATE MATERIALIZED VIEW mvrFournisseurs
REFRESH complete
NEXT sysdate + (1/24/60)
AS
SELECT * FROM cjaverliat.fournisseurs@dbl_eu_n;

--Produits
CREATE MATERIALIZED VIEW mvrPRODUITS
REFRESH fast
NEXT sysdate + (1/24/60)
as
select * from produits;

--Categories
CREATE MATERIALIZED VIEW mvrCATEGORIES
REFRESH FAST
NEXT sysdate + (1/24/60)
AS
SELECT * FROM aouldhamou.categories@DBl_eu_s;

--On verifie qu'on a les mêmes resultats avec Ryori.clients@DBCentrale

--Verifications, si la ligne est vide, c'est ok
SELECT * FROM CLIENTS MINUS (SELECT * FROM Ryori.clients@DBL_principale);
SELECT * FROM COMMANDES MINUS (SELECT * FROM Ryori.COMMANDES@DBL_principale);

--Soucis avec l'ordre des attributs
SELECT * FROM DETAILS_COMMANDES MINUS
(SELECT REF_PRODUIT, NO_COMMANDE, PRIX_UNITAIRE, QUANTITE, REMISE
FROM Ryori.DETAILS_COMMANDES@DBL_principale);

SELECT * FROM EMPLOYES MINUS (SELECT * FROM Ryori.EMPLOYES@DBL_principale);
SELECT * FROM STOCK MINUS (SELECT * FROM Ryori.STOCK@DBL_principale);
SELECT * FROM PRODUITS MINUS (SELECT * FROM Ryori.PRODUITS@DBL_principale);
SELECT * FROM CATEGORIES MINUS (SELECT * FROM Ryori.CATEGORIES@DBL_principale);
SELECT * FROM FOURNISSEURS MINUS (SELECT * FROM Ryori.FOURNISSEURS@DBL_principale);

--suppression des vues materiels pour ne pas que cela tourne dans le vide
drop materialized view mvrproduits;
```

V.D Bilan global des réplications mises en œuvre sur les différents sites

	Europe du Nord	Europe du Sud	Amérique
Réplicat	**Type du réplicat**		
Fournisseurs	x	refresh-complete	refresh-complete
Employes	refresh-fast	refresh-fast	x
Produits	refresh-fast	x	refresh-fast
Categories	refresh-complete	x	refresh-fast

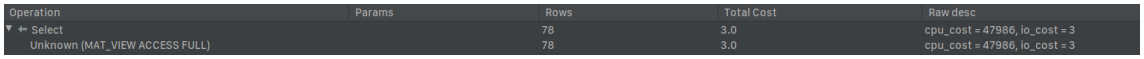
VI Requêtes distribuées : tests et optimisations

VI.A Europe du Nord

On exécute des requêtes pour tester les réplicats, on prendra l'exemple de Produits.
La requête est la suivante :

```
SELECT * FROM PRODUITS;
```

Le résultat est identique à celui sans réplicats (par accès à distance), mais il est bien plus rapide (42ms -> 20ms). Cela montre bien l'intérêt des réplicats pour l'optimisation des requêtes.



Operation	Params	Rows	Total Cost	Raw desc
▼ ← Select		78	3.0	cpu_cost = 47986, io_cost = 3
Unknown (MAT_VIEW ACCESS FULL)		78	3.0	cpu_cost = 47986, io_cost = 3

FIGURE 21 – Plan d'exécution de la requête

VI.B Europe du Sud

VI.B.1 Produits

Pour la réplication de la table Produits stockée localement, il sera nécessaire de créer un log sur celle-ci pour les 2 sites : Europe du nord et Amérique, et leur accorder le droits de lecture sur cette-dernière.

Opérations réalisées en local :

```
--mise en place des logs
CREATE MATERIALIZED VIEW LOG ON PRODUITS;

--mise en place des droits d'acces
GRANT SELECT ON MLOG$_PRODUITS TO CJAVERLIAT;
GRANT SELECT ON MLOG$_PRODUITS TO TBERTHIER;
```

VI.B.2 Catégories

Pour la réplication de la table Catégories stockée localement, il sera nécessaire de créer un log sur celle-ci pour les 2 sites : Europe du nord et Amérique, et leur accorder le droit de lecture sur cette-dernière.

Opérations réalisées en local :

```
--mise en place des logs
CREATE MATERIALIZED VIEW LOG ON CATEGORIES;

--mise en place des droits d'acces
GRANT SELECT ON MLOG$_CATEGORIES TO CJAVERLIAT;
GRANT SELECT ON MLOG$_CATEGORIES TO TBERTHIER;
```

VI.C Amérique