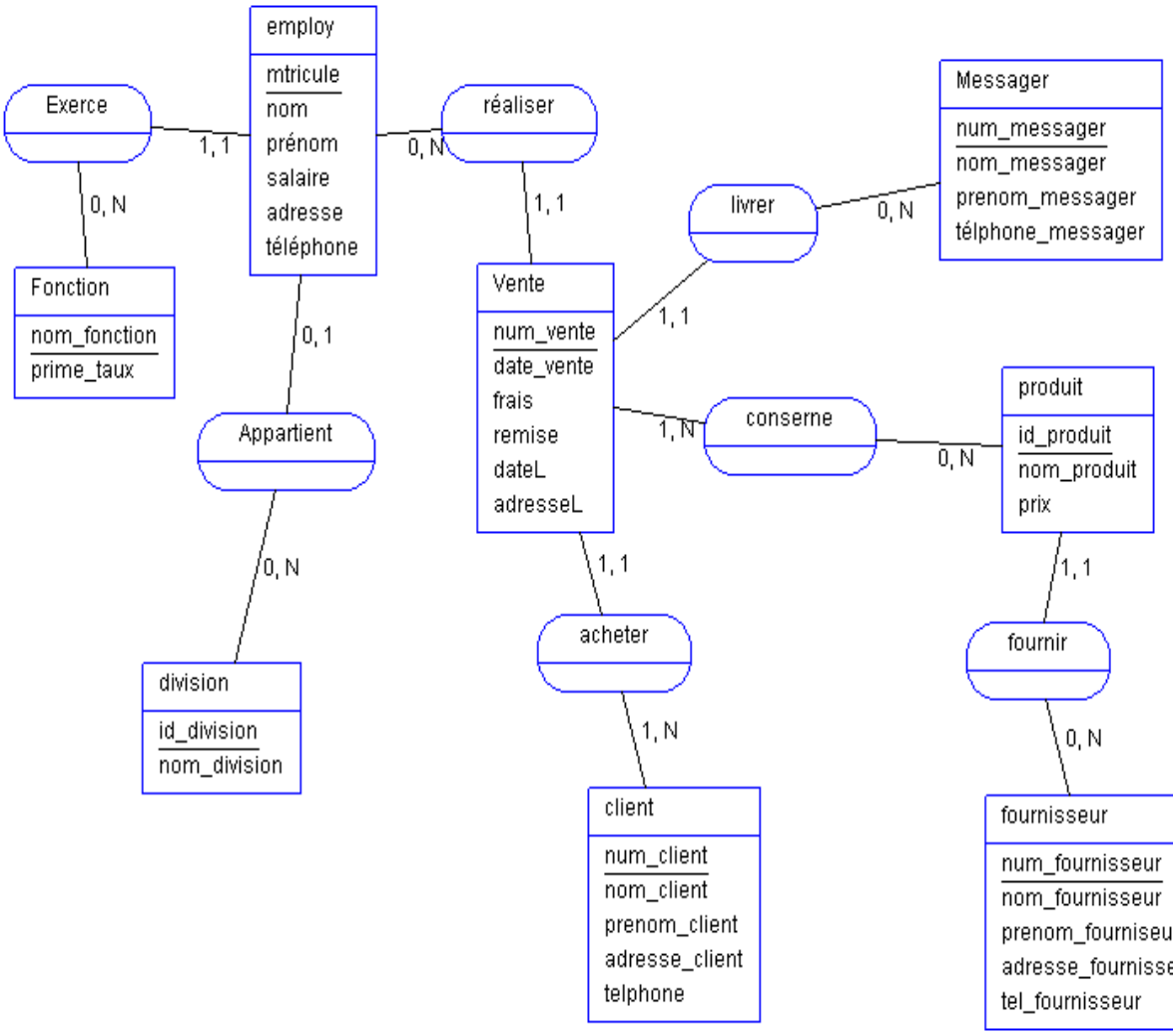
Binôme : HAMRAOUI AKLI **M2 ASS-ACSIS UVSQ 09/10**
BEN HASSEN MONTASSER

Rapport TP BDA

1. TP 1 et 2 Conception BD et Création / Modification de schéma / insertion

1) Après une étude conceptuel des différentes règles de 1 à 10 voila le modèle conceptuel de données choisi et respecté dans l'ensemble des TPs :



2) Création des relations :

(20803317@alsace tild)$ ssh oracle.ens.uvsq.fr Pour accéder a oracle
sqlpluq / pour accéder a sql
sql> @ script .... Pour récupérer le scripte du fichier

```
Drop TABLE DET;
Drop TABLE COM;
Drop TABLE CLI;
Drop TABLE PROD;


CREATE TABLE CLI(CodeCli number(5) Constraint PK_Cli PRIMARY KEY,NomCli
char(20), Pays char(30));

CREATE TABLE PROD(NumProd number(5) Constraint PK_PRO PRIMARY KEY,
NomProd char(20), TypeProd char(10));

CREATE TABLE COM(NumCom number(5) Constraint PK_COM PRIMARY KEY,
CodeCli Number(5) Constraint COM_REF_CLI REFERENCES CLI , FraisPort
number(4),AnCom number(4));

CREATE TABLE DET(
NumCom number(5) Constraint DET_REF_COM REFERENCES COM ,
NumProd number(5) Constraint DET_REF_PRO REFERENCES PROD ,
Qte number(5),
Remise number(5),
Constraint PK_DET PRIMARY KEY (NumCom, NumProd));
```

3) Il y a un ordre à respecter c'est la séquence suivante afin de garantir les contraintes d'intégrité référentielles :
CREATE TABLE CLI -- CREATE TABLE PROD -- CREATE TABLE COM
-- CREATE TABLE DET

4) desc cli : etc...

5) alter table cli modify NomCli varchar(30);

6) alter table cli add Tel number(11);

7) alter table prod add PrixUnit number(11);

8) alter table prod modify(NomProd varchar(20) NOT NULL);

9) insert into cli (CodeCli,NomCli,Pays) values(211,'HAMRAOUI AKLI','France');
insert into prod(NumProd, NomProd, TypeProd)values(104,'Samsungc7','Mobile');

insert into com(NumCom,CodeCli,FraisPort,AnCom)values(14,211,200,14);
insert into det values(14,104,10,6);

10) installation de oracle express

11) update cli set NomCli=UPPER(NomCli);

12) update det set remise=remise*2 where det.numcom=com.numcom and com.ancom>1996

17) Telecharger les fichiers de uvsq creat.sql, cli.ctl etc...puis start CREATE
Pour le chargement il faut quitter sqlplus avec exit puis rentrer dans sqlldr

18) sqlldr / CLI.ctl CLIOUT.log
sqlldr / PRO.ctl PROOUT.log
sqlldr / COM.ctl COM.log
sqlldr / DET.ctl DET.log
sqlldr / FOU.ctl FOU.log

19) Il y a un ordre à respecter **:** CLI.ctl -- PRO.ctl -- COM.ctl -- DET.ctl -- FOU.ctl

20) select count(*) from cli;=91
select count(*) from com;=1076
select count(*) from det;=0
select count(*) from pro;=0
select count(*) from fou;=0

a) Pour com.ctl erreur dans l'insertion
10870;91;72;111997-----AnCom number(4) correction 1997 au lieu de 111997
10934;444;192;1998 ----NumCli Number(5) client 444 n'existe pas correction 44 au lieu de 444

b) Correction supprimer tous les autres tuples du fichier com.ctl puis de laisser cette partie :

```
LOAD DATA
INFILE *APPEND
INTO TABLE COM
FIELDS TERMINATED BY ";" OPTIONALLY ENCLOSED BY ""
(NumCom, NumCli, FraisPort , AnCom)
BEGINDATA
 10934;44;192;1998
10870;91;72;1997
```

2. TP 3 PL/SQL

1) Boucles FOR

```
DECLARE
        x NUMBER := 100;
BEGIN
        FOR i IN 1..10 LOOP
        IF MOD(i,2) = 0 THEN
        INSERT INTO temp VALUES (i,x,'i is even');
        ELSE
        INSERT INTO temp VALUES (i,x,'i is odd');
        END IF;
        x:= x+100;
END LOOP;
COMMIT;
END;
/
```

2) Curseurs

```
DECLARE
        CURSOR c1 is
                SELECT ename,empno,sal FROM emp ORDER BY sal DESC;
        my_ename VARCHAR(32);
        my_empno NUMBER;
        my_sal NUMBER;
        BEGIN
        OPEN c1;
        FOR i IN 1..5 LOOP
                FETCH c1 INTO my_ename,my_empno,my_sal;
                EXIT WHEN c1%NOTFOUND;
                INSERT INTO temp VALUES(my_sal,my_empno,my_ename);
                COMMIT;
        END LOOP;
        CLOSE c1;
END;
/
```

2) PL/SQL versus SQL

```
CREATE TABLE RESULTAT(Code NUMBER,Message VARCHAR(40));

DECLARE
        CURSOR c2 is
                SELECT NUMCLI FROM CLI;
        my_numcli NUMBER;
        my_nombrecom NUMBER;
        BEGIN
        OPEN c2;
        LOOP
                FETCH c2 INTO my_numcli;
                EXIT WHEN c2%NOTFOUND;
                my_nombrecom := TO_NUMBER((SELECT COUNT(NUMCOM)
FROM COM WHERE NUMCLI = my_numcli));
                IF my_nombrecom = 0 THEN
                INSERT INTO Resultat VALUES (my_numcli,'pas de commandes');
                ELSE
                INSERT INTO Resultat VALUES (my_numcli,TO_CHAR('il y a
'||my_nombrecom||' pour le client '||my_numcli));
                END IF;
                COMMIT;
                my_nombrecom := 0;
        END LOOP;
        CLOSE c2;
END;
/
```

3. TP 4 Expérimentations sur la concurrence

1. Préparation de l'expérience

Ajout de l'attribut numérique « Solde » à la table client et insertion du client « Joe » :
alter table cli add solde number(5)
insert into cli(numcli,nomcli) values(5555,'Joe')

2. Niveau d'isolation READ-COMMITED

Q1 : Ouverture de deux fenêtres T1 et T2
1) Suite : Com1, Com2, Solde1, Solde2, Retrait1, Solde2, Rol1, Solde1, Solde2

Transaction 1	**Transaction 2**
Com1	
	Com2
Solde1 (=0)	
	Solde2 (=0)
Retrait1 (-4000)	
	Solde2 (=0)
Rol1 (Annulation=> Solde1 = 0)	
Solde1 (=0)	
	Solde2 (=0)

2) Suite : Com1, Com2, Solde1, Solde2, Retrait1, Solde2, Com1, Solde1, Solde2

Transaction 1	**Transaction 2**
Com1	
	Com2
Solde1 (=0)	
	Solde2 (=0)
Retrait1 (-4000)	
	Solde2 (=0)
Com1 (Solde = -4000)	
Solde1 (=-4000)	
	Solde2 (=-4000)

3) Suite : Com1, Com2, Solde1, Solde2, Retrait1, Solde2, Depot2, Solde1, Solde2, Com1, Com2

Transaction 1	**Transaction 2**
Com1 **Solde1 (=-4000)** **Retrait1 (-8000) Verrou par Update**	**Com2** **Solde2 (=-4000)** **Solde2 (=-4000)** **Depot2 (= +6000)** **Blocage Update** **Blocage impossible causé par le verrou**

Q2 : ORACLE ne garantie pas la sérialisation des transactions prenons l'exemple de la séquence 2 :
Com1, Com2, Solde1 (0), Solde2 (0), Retrait1 (-4000), Solde2 (0), Com1, Solde1 (-4000), Solde2 (- 4000)
Sérialisation de la séquence =>

1) Com1, Solde1 (0), Retrait1 (-4000), Com1, Solde1 (-4000)
2) Com2, Solde2 (-4000 ≠ 0), Solde2 (-4000 ≠ 0), Solde2 (-4000 ≠ 0)

Nous remarquons que Solde2 est passé de 0 à dans le test de sérialisation donc oracle ne garantie pas la sérialisation.

Q3 : Lors d'une lecture un verrou court est placé durant la lecture, nous pouvons dire qu'il n' y a pas de verrou car ce dernier est court afin de respecter en théorie le READ-COMMITED

Q4 : Le comportement correspond au niveau d'isolation READ-COMMITED de la norme SQL on effet ORACLE respecte le principe en ajoutant des options telles que le placement d'un verrou court pour la lecture.

4. TP 5 Optimisation

Q1) Avec index :

Nous créons un fichier CLEAN.sql contenant uniquement la création des tables suivantes CLI, FOU, COM, PRO et les insertions sans la table DET et ses insertions pour respecter les contraintes d'intégrité référentielles:

```
Drop TABLE PRO;
Drop TABLE COM;
Drop TABLE FOU;
Drop TABLE CLI;

CREATE TABLE CLI(
  NumCli number(5) Constraint PK_Cli PRIMARY KEY,
  NomCli char(20),
  Pays char(30),
  Tel char(15));

CREATE TABLE FOU(
  NumFou number(2) Constraint PK_FOU PRIMARY KEY,
  NomFou char(20),
  Pays char(30),
  Tel char(15));

CREATE TABLE COM(
  NumCom number(5) Constraint PK_COM PRIMARY KEY,
  NumCli Number(5) Constraint COM_REF_CLI REFERENCES Cli ON DELETE
CASCADE,
  FraisPort number(4),
  AnCom number(4));

CREATE TABLE PRO(
  NumPro number(5) Constraint PK_PRO PRIMARY KEY,
  NumFou number(2) Constraint PRO_REF_FOU REFERENCES Fou ON DELETE
CASCADE,
  NomPro char(20),
  TypePro char(10),
  PrixUnit number(3));
insert into CLI values (1,'Maria','Pologne','4867427275');
…
insert into COM values (10000,54,26,1999);
…
insert into FOU values (1,'Exotic','Allemagne','1019168842');
…
insert into PRO values (1,1,'Chai','A',90);
```

Sans les insertions de la table DET

b) Nous créons un fichier DET1 .sql contenant la création de la table DET et une procédure sql pour insérer les tuples ayant la structure suivante :

```
Drop TABLE DET;

CREATE TABLE DET(
  NumCom number(5) Constraint DET_REF_COM REFERENCES Com ON DELETE
CASCADE,
  NumPro number(5)Constraint DET_REF_PRO REFERENCES Pro ON DELETE
CASCADE,
  Qte number(5),
  Remise number(5),
  Constraint PK_DET PRIMARY KEY (NumCom, NumPro));

Set timing on
Declare
Begin
Insertion des tuples de la table DET
…
End;
/
```

<u>Résultat avec index</u>: Temps d'éxecution 04s:20ms

2) Sans index :

Nous créons un fichier createnoind.sql contenant uniquement la création des tables suivantes CLI, FOU, COM, PRO et les insertions sans la table DET et ses insertions.

```
Drop TABLE PRO;
Drop TABLE COM;
Drop TABLE FOU;
Drop TABLE CLI;

CREATE TABLE CLI(
  NumCli number(5) ,
  NomCli char(20),
  Pays char(30),
  Tel char(15));

CREATE TABLE FOU(
  NumFou number(2) ,
  NomFou char(20),
  Pays char(30),
  Tel char(15));
```

```
CREATE TABLE COM(
  NumCom number(5) ,
  NumCli Number(5) ,
  FraisPort number(4),
  AnCom number(4));

CREATE TABLE PRO(
  NumPro number(5) ,
  NumFou number(2),
  NomPro char(20),
  TypePro char(10),
  PrixUnit number(3));
```

b) Nous créons un fichier DET1 .sql contenant la création de la table DET et une procédure sql pour insérer les tuples ayant la structure suivante :

```
Drop TABLE DET;

CREATE TABLE DET(
  NumCom number(5) ,
  NumPro number(5),
  Qte number(5),
  Remise number(5));

Set timing on
Declare
Begin
Insertion des tuples de la table DET
…
End;
/
```

<u>Résultat sans index</u>: Temps d'exécution 07s:30ms

On remarque que les indexes réduit le temps de l'insertion.
Q2) Sans index

R1 : SELECT count(distinct L.numcli)from CLI L,COM C,DET D,PROD P
WHERE L.numCli=C.numcli and C.NumCom=D.NumCom and D.NumPro=P.NumProd;

```
COUNT(DISTINCTL.NUMCLI)
-----------------------
                      0

Ecoule : 00 :00 :00.00

Plan d'execution
```

```
------------------------------------------------------------
Plan hash value: 743381048

-------------------------------------------------------------------------------
---------
---------

| Id  | Operation                     | Name   | Rows  | Bytes | Cost
(%CPU)| T
ime     |

-------------------------------------------------------------------------------
---------
---------

|   0 | SELECT STATEMENT              |        |     1 |    78 |     9
(12)| 0
0:00:01 |

|   1 |  SORT GROUP BY                |        |     1 |    78 |
|
        |

|   2 |   NESTED LOOPS                |        |     1 |    78 |     9
(12)| 0
0:00:01 |

|   3 |    NESTED LOOPS               |        |     1 |    65 |     9
(12)| 0
0:00:01 |

|*  4 |     HASH JOIN                 |        |     1 |    39 |     8
(13)| 0
0:00:01 |

|   5 |      TABLE ACCESS FULL        | PROD   |     1 |    13 |     2
(0)| 0
0:00:01 |

|   6 |      TABLE ACCESS FULL        | DET    |  2809 | 73034 |     5
(0)| 0
0:00:01 |

|   7 |     TABLE ACCESS BY INDEX ROWID| COM    |     1 |    26 |     1
(0)| 0
0:00:01 |

|*  8 |      INDEX UNIQUE SCAN        | PK_COM |     1 |       |     0
(0)| 0
0:00:01 |

|*  9 |    INDEX UNIQUE SCAN          | PK_CLI |     1 |    13 |     0
(0)| 0
0:00:01 |

-------------------------------------------------------------------------------
---------
```

```
---------


Predicate Information (identified by operation id):
---------------------------------------------------

   4 - access("D"."NUMPRO"="P"."NUMPROD")
   8 - access("C"."NUMCOM"="D"."NUMCOM")
   9 - access("L"."NUMCLI"="C"."NUMCLI")

Note
-----
   - dynamic sampling used for this statement
```