

Mamadou SIDIBE

Basma HAOUZI

IDSCC3

Projet à réaliser : Module Bases de données Avancées

1. Proposition un cahier de charge d'une application de gestion dans le domaine de la vente de produits pour une entreprise :

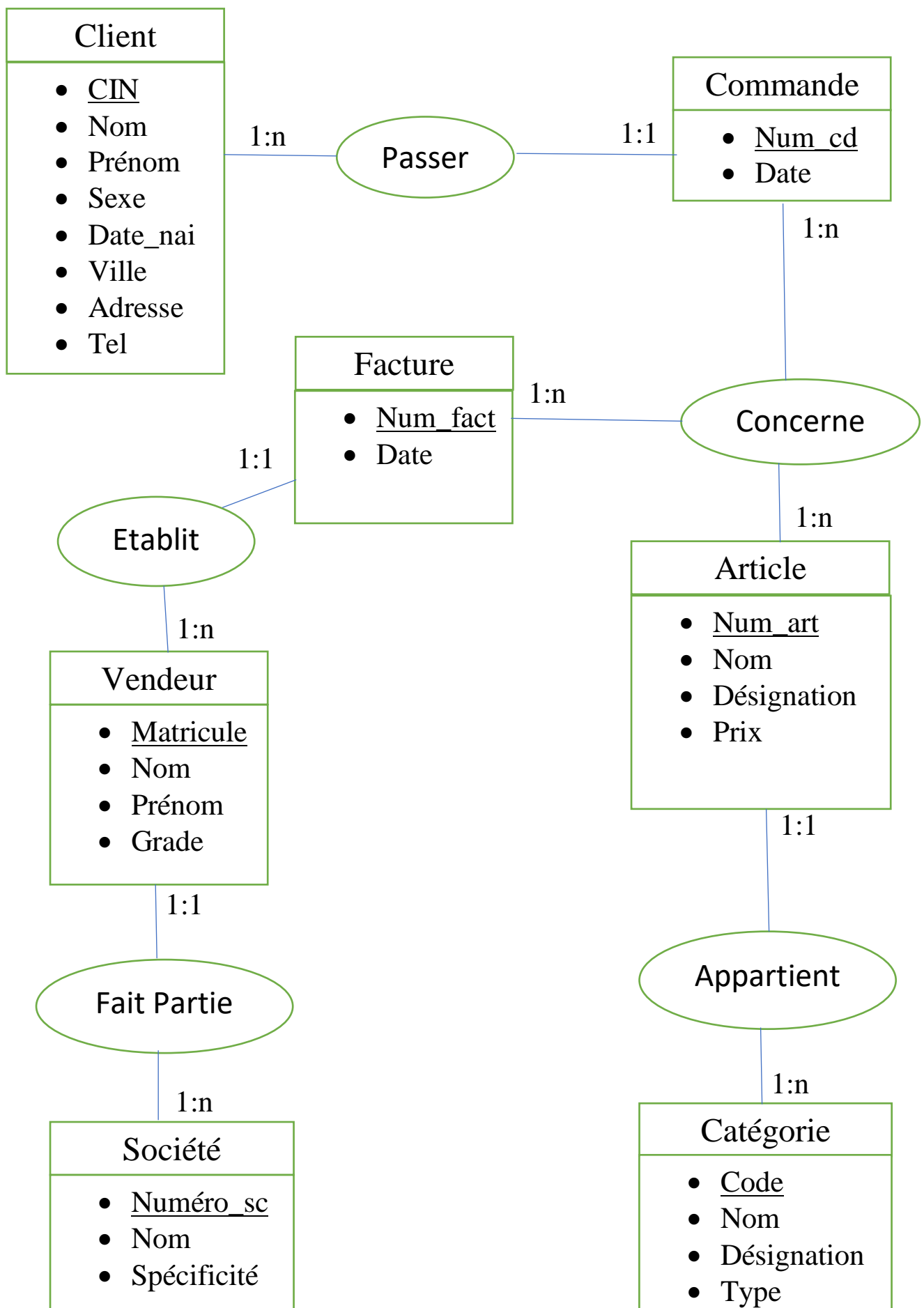
Soit une entreprise désirant faire de la gestion des ventes de ces Articles ; **Un client** est défini par son nom, son prénom, un cin, un sexe, une date de naissance, une ville, une adresse et un numéro de téléphone ; il peut passer une ou plusieurs commandes ; **Une commande** est définie par un numéro de commande et une date de commande.

Une commande peut concerner un ou plusieurs Article et une facture qui est établie par un vendeur. **Un Article** est défini par un numéro de série, un nom, une désignation, et un prix ; **Une facture** est définie par un numéro et une date.

Un article appartient à **une Catégorie** définie par un code, un nom une désignation et un type.

Une facture est établie est établie par un vendeur et chaque vendeur appartient à une société de l'entreprise de l'entreprise. **Un vendeur** est défini par un matricule, un nom, un prénom, et un grade ; Chaque **Société** à un nom, un numéro et une spécificité.

2. Définir le modèle conceptuel de données de données (MCD) correspondant à mon application :



3. Créer sur le SGBD Oracle les tables qui correspondent à ce modèle conceptuel de données :

Modèle conceptuel de données (MCD) #####
3- Création des tables qui correspondent à mon modèle conceptuel de données :

```
CREATE TABLE client
(
    cin varchar(20) PRIMARY KEY NOT NULL,
    nom varchar(50),
    prenom varchar(50),
    sexe varchar(1),
    date_nai Date,
    ville varchar(50),
    adresse varchar(30),
    tel varchar(50)
);

CREATE TABLE commande
(
    num_cd int,
    cin varchar(20),
    date_cd date,
    PRIMARY KEY (cin, num_cd),
    unique(num_cd),
    FOREIGN KEY (cin) REFERENCES client(cin)
);
```

```
CREATE TABLE societe
(
    num_sc int,
    nom varchar(50),
    spe varchar(250),
    PRIMARY KEY (num_sc)
);
```

```
CREATE TABLE categorie
(
```

```

        code int,
        nom varchar(50),
        designation varchar(250),
        type_categorie varchar(2),
        PRIMARY KEY (code)

);

CREATE TABLE article
(
    num_art int,
    nom varchar(50),
    designation varchar(50),
    categ int,
    prix int,
    PRIMARY KEY (num_art,categ),
    unique(num_art),
    FOREIGN KEY (categ) REFERENCES categorie(code)
);

CREATE TABLE vendeur
(
    matricule varchar(10),
    nom varchar(50),
    prenom varchar(50),
    grade varchar(30),
    num_sc int,
    PRIMARY KEY (matricule ,num_sc),
    unique(matricule),
    FOREIGN KEY (num_sc) REFERENCES societe(num_sc)
);

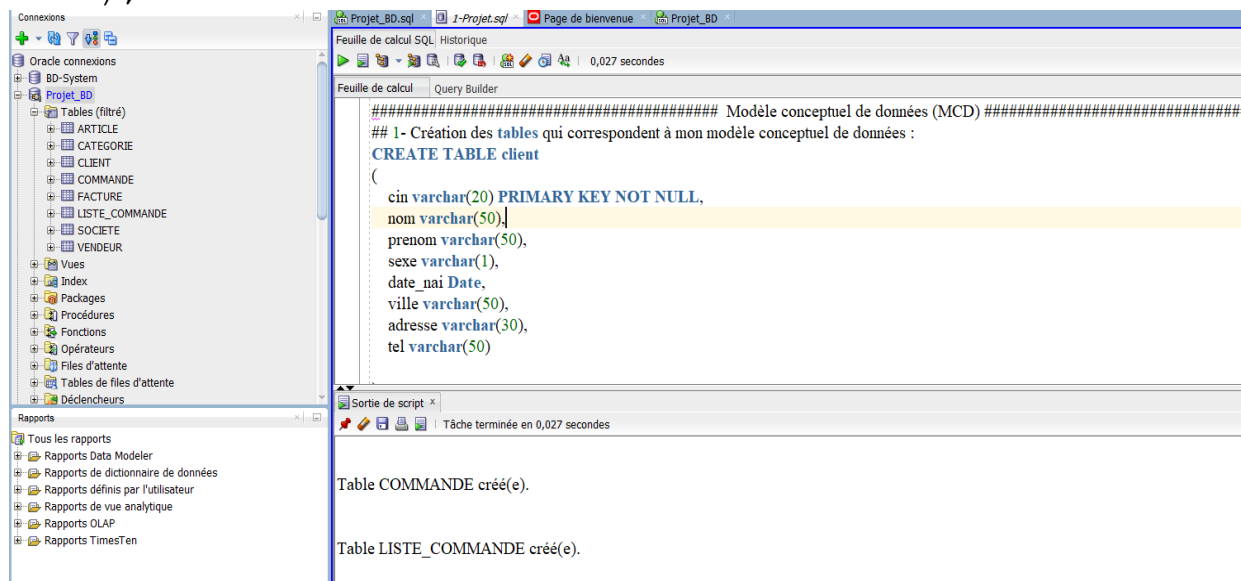
CREATE TABLE facture
(
    num_facture int,
    date_facture date,
    mat varchar(10),
    PRIMARY KEY (num_facture,mat),
    unique(num_facture),
    FOREIGN KEY (mat) REFERENCES vendeur(matricule)
);

```

```

CREATE TABLE liste_commande
(
    num_facture int,
    num_art int,
    num_cd int,
    PRIMARY KEY(num_facture, num_art,num_cd),
    FOREIGN KEY (num_facture) REFERENCES
facture(num_facture),
    FOREIGN KEY(num_art) REFERENCES article(num_art),
    FOREIGN KEY(num_cd) REFERENCES commande(num_cd)
);

```



4. Alimenter ces tables par des jeux de données :

4- Alimenter ces tables par des jeux de données

##Insertion dans la table client

```

INSERT INTO client VALUES
('A01213','SIDIBE','Mamadou','M','16/02/02','Oujda','
HAY Qods Rue 25 Porte 1','06xxxxxxxxxxxx');
INSERT INTO client VALUES
('B018957','KEITA','Kissima','M','18/09/00','Casa','H
AY Rabi Rue 75 Porte 7','07xxxxxxxxxxxx');

```

```
INSERT INTO client VALUES
('C04142','DIALLO','Fatoumata','F','25/03/99','Rabat'
,'Doha Rue 02 Porte 5','05xxxxxxxxxxx');
```

```
##Instertion dans la table commande
INSERT INTO commande VALUES (16,'A01213','16/02/22');
INSERT INTO commande VALUES
(18,'B018957','30/01/22');
INSERT INTO commande VALUES (25,'C04142','10/04/22');
```

```
##Instertion dans la table facture
INSERT INTO facture VALUES (01,'16/02/22','FX0648');
INSERT INTO facture VALUES (02,'30/01/22','FX0648');
INSERT INTO facture VALUES (04,'10/04/22','SR0489');
```

```
##Insertion dans la table categorie
INSERT INTO categorie VALUES (3,'Soins et
Beauté','Produits de soins debeaute pour le visage et
le corps de très bonne qualité','B');
INSERT INTO categorie VALUES
(2,'Musculatation','Produits pour la pratique dexercice
physique et de sport a la maison et dehors','B');
INSERT INTO categorie VALUES
(1,'Alimentaire','Produits alimentaire de toute sorte
frais et de premiere qualité','A');
```

```
##Insertion dans la table societe
INSERT INTO societe VALUES (201,'Alpha','Premiere
division de lentreprise pour la gestion de tout type
de vente et premiere tete daffiche de la societe');
INSERT INTO societe VALUES (864,'Beta','spécialisaer
dans la vente de produits alimentaire mais aussi tres
polyvalent dans les autres dommines');
```

```
##Insertion dans la table vendeur
INSERT INTO vendeur VALUES
('FX0648','DIALLO','Ahmed','Grade 3',201);
INSERT INTO vendeur VALUES
('SR0489','SIDIBE','Youssouf','Grade 2',864);
```

```
##Insertion dans la table article
INSERT INTO article VALUES (16,'Parfun Dior
Sauvage','Parfum pour homme de la marque Dior
',3,35);
INSERT INTO article VALUES (78,'Altères
50kg','Altèeres de musculation au nombre de 5 par
kit',2,300);
INSERT INTO article VALUES (25,'Riz Blanc','Sac de
riz blanc de 25kg ',1,150);
INSERT INTO article VALUES (07,'Spagetti','1kg de
pate fraiche 1er choix ',1,12);
```

```
##Insertion dans la table liste_commande
INSERT INTO liste_commande VALUES (01,16,16);
INSERT INTO liste_commande VALUES (02,16,18);
INSERT INTO liste_commande VALUES (02,78,18);
INSERT INTO liste_commande VALUES (04,25,25);
```

5-Ecrire une requête permettant d'appliquer la jointure Left Join.

Commenter cette requête :

```
##5-Ecrire une requête permettant d'appliquer la
jointure Left Join Commenter cette requête :
select
client.cin,client.nom,client.prenom,commande.num_cd,l
iste_commande.num_facture,article.nom as nom_article
from client
left join commande on client.cin=commande.cin
left join  liste_commande on
commande.num_cd=liste_commande.num_cd
left join article on
article.num_art=liste_commande.num_art;
```

##5-Ecrire une requête permettant d'appliquer la jointure **Left Join** Commenter cette requête :

```
select client.cin,client.nom,client.prenom,commande.num_cd,liste_commande.num_facture,article.nom as nom_article from client
left join commande on client.cin=commande.cin
left join  liste_commande on commande.num_cd=liste_commande.num_cd
left join article on article.num_art=liste_commande.num_art;
```

Sortie de script x

Résultat de requête x

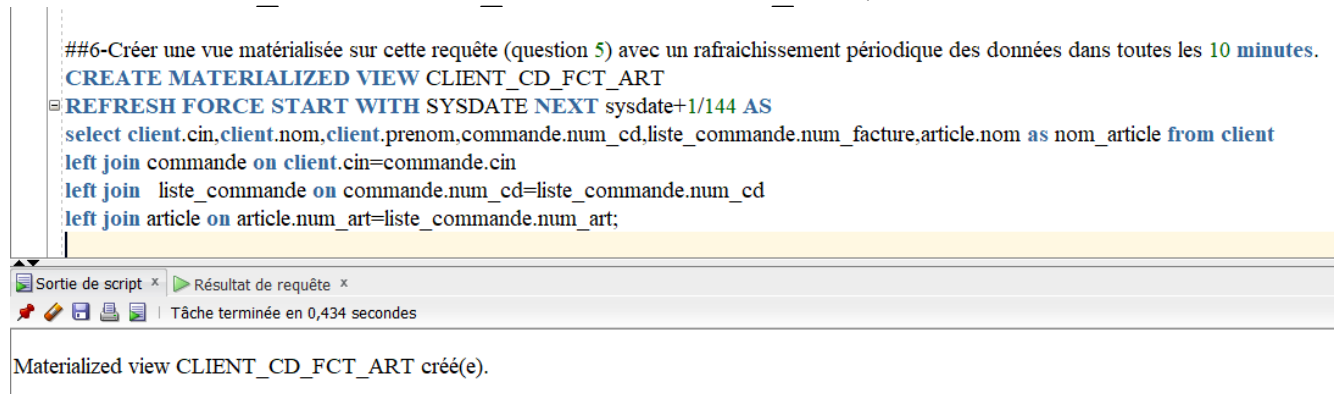
Toutes les lignes extraites : 4 en 0,006 secondes

CIN	NOM	PRENOM	NUM_CD	NUM_FACTURE	NOM_ARTICLE
1 A01213	SIDIBE	Mamadou	16		1 Parfun Dior Sauvage
2 B018957	KEITA	Kissima	18		2 Parfun Dior Sauvage
3 B018957	KEITA	Kissima	18		2 Altères 50kg
4 C04142	DIALLO	Fatoumata	25		4 Riz Blanc

Cette requête retourner le cin, le nom, le prénom des **clients** ainsi que le numéro leur de commande, le numéro de leur de la facture et le nom de l'article qu'ils ont commandé.

6-Créer une vue matérialisée sur cette requête (question 5) avec un rafraichissement périodique des données dans toutes les 10 minutes. Expliquer l'intérêt de l'utilisation de l'option 'rafraichissement périodique' ;

```
##6-Créer une vue matérialisée sur cette requête
(question 5) avec un rafraichissement périodique des
données dans toutes les 10 minutes.
CREATE MATERIALIZED VIEW CLIENT_CD_FCT_ART
REFRESH FORCE START WITH SYSDATE NEXT sysdate+1/144
AS
select
client.cin,client.nom,client.prenom,commande.num_cd,l
iste_commande.num_facture,article.nom as nom_article
from client
left join commande on client.cin=commande.cin
left join  liste_commande on
commande.num_cd=liste_commande.num_cd
left join article on
article.num_art=liste_commande.num_art;
```



L'utilisation de l'option rafraichissement périodique permet a la vues matérialiser de mettre a jour ces donne d'une manière périodique et donc rester dans un état coherent avec la base de donnée

7-Créer un Index simple sur trois colonnes d'une table quelconque faisant partie de ce MCD;

##7- Créer un Index simple sur la colonne nom et prénom de la table client et la colonne nom de la table article;

```
CREATE INDEX index_NOM_PRENOM_CLIENT on  
client(nom,prenom);
```

```
CREATE INDEX index_NOM_ARTICLE on article(nom);
```

##7- Créer un **Index** simple sur la colonne nom et prénom de la **table client** et la colonne nom de la **table article**;
CREATE INDEX index_NOM_PRENOM_CLIENT **on** client(nom,prenom);
CREATE INDEX index_NOM_ARTICLE **on** article(nom);

Sortie de script x Résultat de requête x
Tâche terminée en 0,033 secondes

Index INDEX_NOM_PRENOM_CLIENT créé(e).

Index INDEX_NOM_ARTICLE créé(e).

8-Expliquer en se basant sur un exemple (extrait à partir de ce MCD) l'objectif de l'utilisation des transactions :

##8-Exemple de transaction basé sur le MCD ;

```
BEGIN
```

```
UPDATE article SET prix =125 where num_art=25;
```

```
UPDATE vendeur SET grade='GRADE X' where  
matricule='FX0648';
```

```
END;
```

##8-Exemple de **transaction** basé sur le MCD ;

BEGIN

UPDATE article **SET** prix =125 **where** num_art=25;

UPDATE vendeur **SET** grade='GRADE X' **where** matricule='FX0648';

END;

COMMIT;

Sortie de script x
Tâche terminée en 0,07 secondes

Procédure PL/SQL terminée.

Validation (commit) terminée.

Dans l'exemple ci-dessus nous avons utilisé une transaction pour effectuer plusieurs mises à jour sur plusieurs tables en un seul bloc d'instruction et cette transaction nous permet d'assurer en cas de panne que notre base de données restera dans un état cohérent.

9-Lancer deux transactions concurrentes et définir des scénarios de verrouillage partagé et de verrouillage exclusif. Par la suite, commenter chaque scénario ;

```
-----  
##Scénarios de verrouillage partagé  
-----scénario 1: verrouillage partagé---  
-----  
--TRANSACTION T1:  
SET TRANSACTION NAME 'T1';  
  
--TRANSACTION T2:  
SET TRANSACTION NAME 'T2';  
  
--TRANSACTION T1:  
LOCK TABLE client IN SHARE MODE; -- Verrouiller la  
table entière client, Pour ne pas lire des valeurs  
impropres ;  
  
--TRANSACTION T2:  
SELECT cin,nom FROM client;  
update client set ville='Tanger' where cin='A01213';  
---BLOCAGE  
  
--TRANSACTION T1:  
ROLLBACK;  
  
--TRANSACTION T2:  
--Vérifier la session T2  
--Déblocage, Instruction Effectuer  
ROLLBACK;
```

```

##7- Créer un Index simple sur la colonne nom et prénom de la table client et la colonne nom de la table article;
CREATE INDEX index_NOM_PRENOM_CLIENT on client(nom, prenom);
CREATE INDEX index_NOM_ARTICLE on article(nom);

##8-Exemple de transaction basé sur le MCD ;
BEGIN
UPDATE article SET prix =125 where num_art=25;
UPDATE vendeur SET grade='GRADE X' where matricule='F3';
END;

COMMIT;

##9
SET TRANSACTION NAME 'T1';
LOCK TABLE client IN SHARE MODE;

```

Invite de commandes - sqlplus

Connecté à :

Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production

Version 19.3.0.0.0

SQL> SET TRANSACTION NAME 'T2';

Transaction définie.

SQL> select cin,nom from client;

CIN	NOM
A01213	SIDIBE
B018957	KEITA
C04142	DIALLO

SQL> update client set ville='Tanger' where cin='A01213';

Sortie de script x

Tâche terminée en 0,065 secondes

Succès de l'élément transaction NAME.

Succès de l'élément Lock.

Après fin de la transaction T1 et donc fin du verrouillage :

```

##8-Exemple de transaction basé sur le MCD ;
BEGIN
UPDATE article SET prix =125 where num_art=25;
UPDATE vendeur SET grade='GRADE X' where matricule='F3';
END;

COMMIT;

##9
SET TRANSACTION NAME 'T1';
LOCK TABLE client IN SHARE MODE;
ROLLBACK;

```

Invite de commandes - sqlplus

Connecté à :

Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production

Version 19.3.0.0.0

SQL> SET TRANSACTION NAME 'T2';

Transaction définie.

SQL> select cin,nom from client;

CIN	NOM
A01213	SIDIBE
B018957	KEITA
C04142	DIALLO

SQL> update client set ville='Tanger' where cin='A01213';

1 ligne mise à jour.

SQL> ROLLBACK;

Annulation (rollback) effectuée.

SQL>

Sortie de script x

Tâche terminée en 0,03 secondes

Succès de l'élément Lock.

Annulation (rollback) terminée.

```

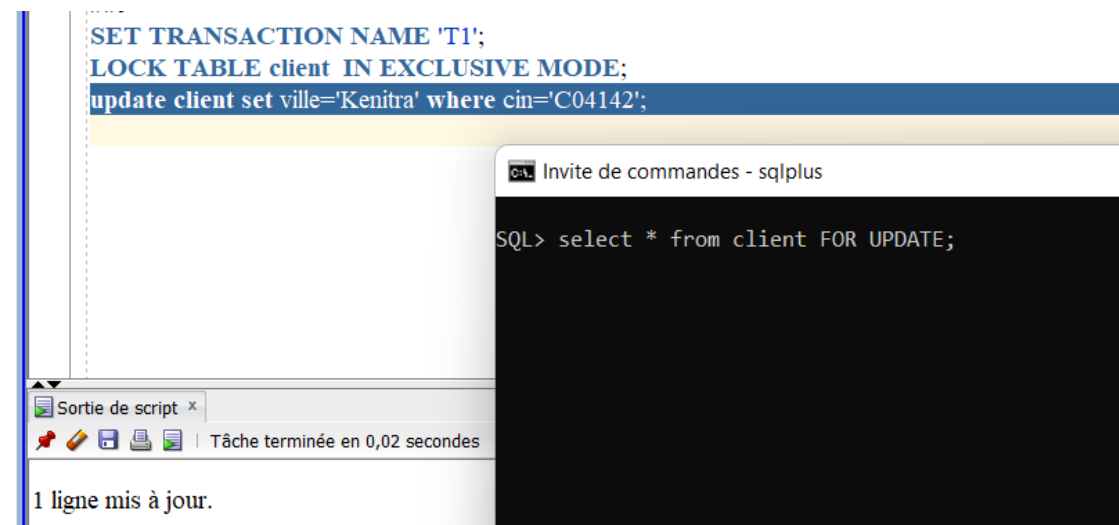
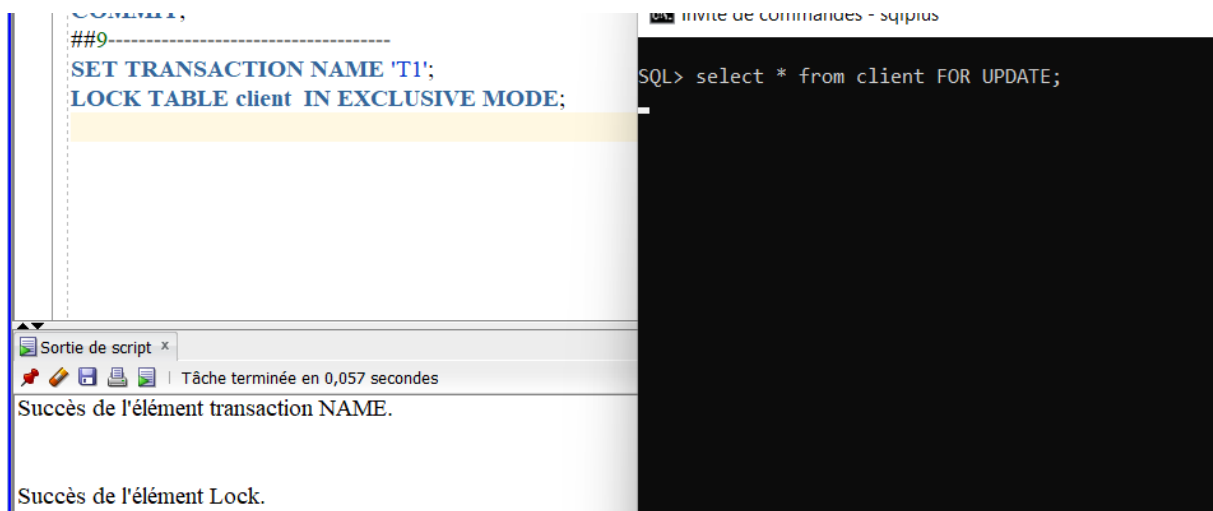
##Scénario de verrouillage exclusif.
-----scénario 3: verrouillage exclusif
--TRANSACTION T1:
LOCK TABLE client IN EXCLUSIVE MODE; ---Or Update
Projet...

--TRANSACTION T2:
select * from client FOR UPDATE; -- Blocage

--TRANSACTION T1:
update client set ville='Kenitra' where cin='C04142';
commit;

--TRANSACTION T2:
--Vérifier la session T2
--Requête exécuter

```



COMMIT;

##9-----

SET TRANSACTION NAME 'T1';

LOCK TABLE client IN EXCLUSIVE MODE;

update client set ville='Kenitra' where cin='C04142';

commit;

Sortie de script

Tâche terminée en 0,047 secondes

1 ligne mis à jour.

Validation (commit) terminée.

Invite de commandes - sqlplus

```
-----
C04142          DIALLO          F 25/03/99
Fatoumata
Kenitra
```

```
-----
CIN            NOM
```

```
-----
PRENOM          S DATE_NAI
```

```
-----
VILLE
```

```
-----
ADRESSE
```

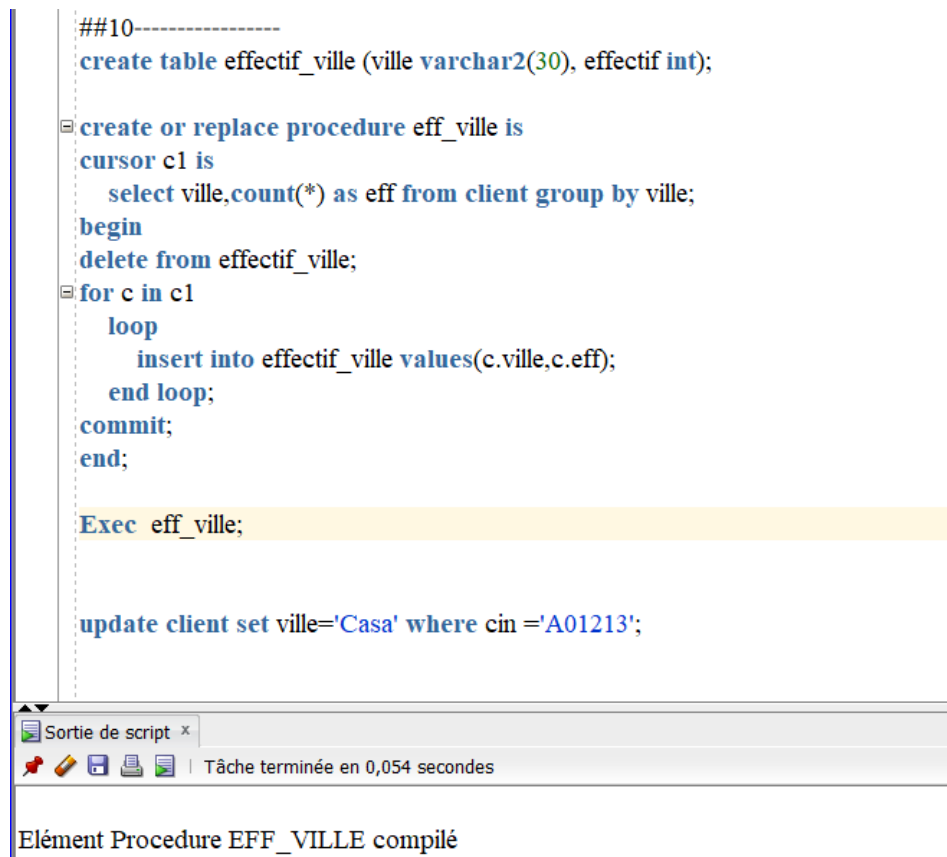
```
-----
TEL
```

```
-----
Doha Rue 02 Porte 5
05xxxxxxxxxxxx
```

```
SQL> ^S_
```

10- Ecrire une procédure stockée PL/SQL pour l'exécution d'un traitement sur ces tables, en utilisant les curseurs ;

```
-----  
##10-----  
create table effectif_ville (ville varchar2(30),  
effectif int);  
  
create or replace procedure eff_ville is  
cursor c1 is  
    select ville,count(*) as eff from client group by  
ville;  
begin  
delete from effectif_ville;  
for c in c1  
    loop  
        insert into effectif_ville  
values(c.ville,c.eff);  
    end loop;  
commit;  
end;  
Exec  eff_ville;
```



```
##10-----  
create table effectif_ville (ville varchar2(30), effectif int);  
  
create or replace procedure eff_ville is  
cursor c1 is  
    select ville,count(*) as eff from client group by ville;  
begin  
delete from effectif_ville;  
for c in c1  
    loop  
        insert into effectif_ville values(c.ville,c.eff);  
    end loop;  
commit;  
end;  
  
Exec  eff_ville;
```

Sortie de script x

Tâche terminée en 0,054 secondes

Elément Procedure EFF_VILLE compilé

11- Créer un trigger de ligne, permettant de garder l'historique de données modifiées ou supprimées. Ce trigger sera déclenché automatiquement après chaque modification ou suppression d'un enregistrement d'une table donnée (choisir une seule table) ;

```
-----  
##11-----  
CREATE TABLE historiques_article  
(  
  
    date_op date,  
    num_art int,  
    nom varchar(50),  
    designation varchar(50),  
    categ int,  
    prix int  
  
);  
  
CREATE OR REPLACE TRIGGER historiqueArticle  
AFTER DELETE OR UPDATE ON article FOR EACH ROW  
BEGIN  
    IF DELETING OR UPDATING THEN  
        INSERT INTO historiques_article VALUES  
(SYSDATE, :OLD.num_art, :OLD.nom, :OLD.designation,  
:OLD.categ, :OLD.prix);  
    END IF;  
END;
```

```

##11-----
CREATE TABLE historiques_article
(
    date_op date,
    num_art int,
    nom varchar(50),
    designation varchar(50),
    categ int,
    prix int
);

CREATE OR REPLACE TRIGGER historiqueArticle
AFTER DELETE OR UPDATE ON article FOR EACH ROW
BEGIN
    IF DELETING OR UPDATING THEN
        INSERT INTO historiques_article VALUES (SYSDATE, :OLD.num_art, :OLD.nom, :OLD.designation, :OLD.categ, :OLD.prix);
    END IF;
END;

```

Sortie de script x
 Tâche terminée en 0,046 secondes

Elément Trigger HISTORIQUEARTICLE compilé

-Après modification sur la table article :

```
update article set prix=145 where num_art=25;
```

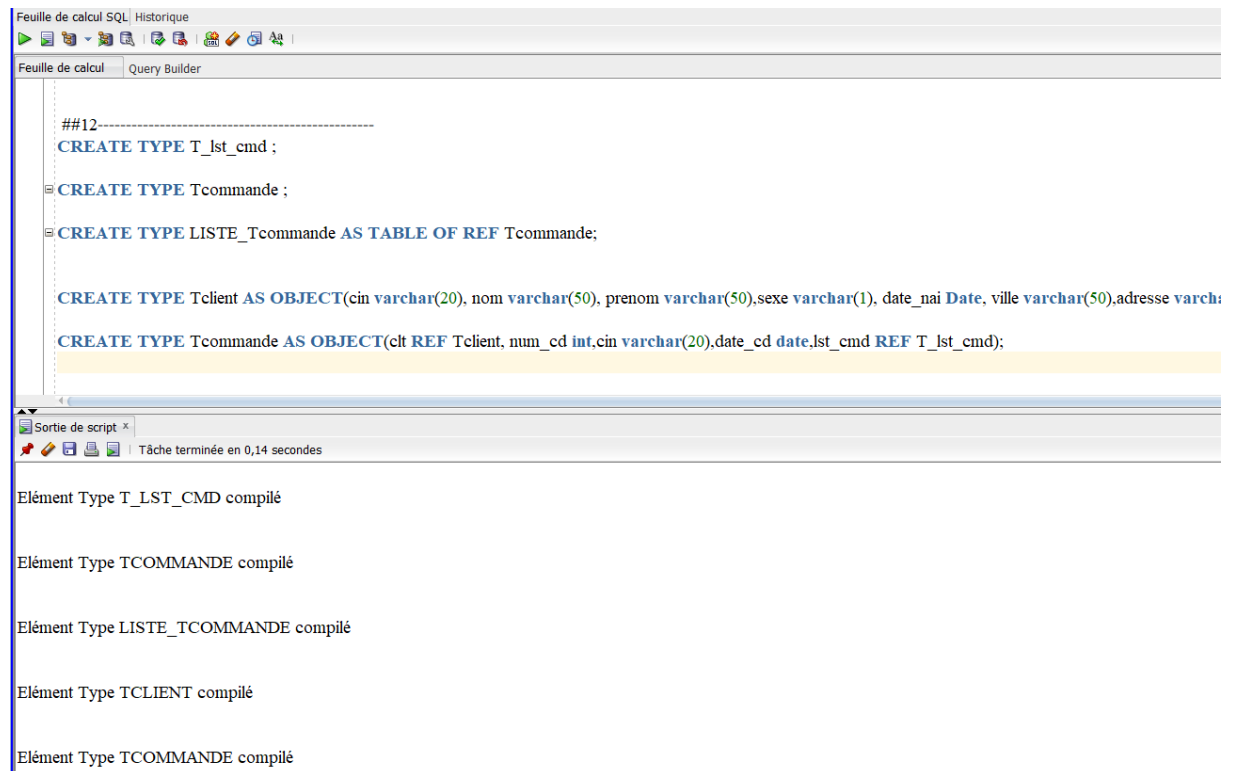
1 ligne mis à jour.

Colonne	Données	Model	Contraintes	Droits	Statistiques	Déclencheurs	Flashback	Dépendances	Dét
DATE_OP	NUM_ART	NOM	DESIGNATION	CATEG	PRIX				
1 07/05/22	25	Riz Blanc	Sac de riz blanc de 25kg	1	125				

12-On souhaite représenter ce modèle conceptuel de données (MCD) défini dans la question 2, sous forme des objets et tables en adoptant le modèle relationnel objet. Créer les types d'objet et les tables d'objets qui correspondent à ce MCD en combinant entre les associations symétriques et d'agrégation (Justifier votre choix) ;

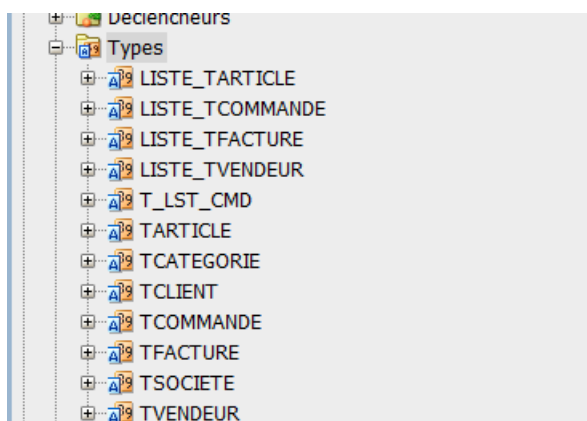
##12-----

- CREATE TYPE `T_lst_cmd` ;
- CREATE TYPE `Tcommande` ;
- CREATE TYPE `LISTE_Tcommande` AS TABLE OF REF `Tcommande`;
- CREATE TYPE `Tclient` AS OBJECT(`cin` varchar(20), `nom` varchar(50), `prenom` varchar(50), `sexe` varchar(1), `date_nai` Date, `ville` varchar(50), `adresse` varchar(30), `tel` varchar(50), `cmd` `LISTE_Tcommande`);
- CREATE TYPE `Tcommande` AS OBJECT(`clt` REF `Tclient`, `num_cd` int, `cin` varchar(20), `date_cd` date, `lst_cmd` REF `T_lst_cmd`);



-
- CREATE TYPE Tfacture;
 - CREATE TYPE Tsociete;
 - CREATE TYPE LISTE_Tfacture AS TABLE OF REF Tfacture ;
 - CREATE TYPE Tvendeur AS OBJECT(factures LISTE_Tfacture, sct REF Tsociete, matricule varchar(10), nom varchar(50), prenom varchar(50), grade varchar(30), num_sc int);
 - CREATE TYPE Tfacture AS OBJECT(vds REF Tvendeur, num_facture int, date_facture date, mat varchar(10), lst_cmd REF T_lst_cmd);
 - CREATE TYPE LISTE_Tvendeur AS TABLE OF REF Tvendeur ;

- CREATE TYPE TSociete AS OBJECT(vendeurs LISTE_
Tvenduer,num_sc number,nom varchar(50), spe
varchar(250));
-
- CREATE TYPE Tarticle ;
 - CREATE TYPE LISTE_ Tarticle AS TABLE OF REF
Tarticle ;
 - CREATE TYPE Tcategorie AS OBJECT(articles LISTE_
Tarticle, code int,nom varchar(50),designation
varchar(250),type_categorie varchar(2));
 - CREATE TYPE Tarticle AS OBJECT(ctg REF Tcategorie,
num_art int,nom varchar(50),designation
varchar(50),categ int,prix int, lst_cmd REF
T_lst_cmd);
-
- ```
CREATE TYPE T_lst_cmd AS OBJECT (cmd
LISTE_Tcommande,art LISTE_ Tarticle,fct LISTE_
Tfacture);
```



Nous avons utilisé l'association symétrie pour la création de ces type(client/commande) car client n'est pas prédominant par rapport à commande de même pour les associations (société/vendeur/facture) et (catégorie/article).

