

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique



Université des Sciences et de la Technologie Houari Boumediene

Faculté d'Electronique et d'Informatique

Département Informatique

Filière : Informatique

Rapport Base de données avancées

Projet n°01SQL3-Oracle

Réalisé par :

Benterzi Mohamed El Amine
BESSAI Zoheir

181831068835
181831068779

Sommaire

PARTIE I : Modélisation orientée objet	3
PARTIE II : Création des TablesSpaces et utilisateur	4
Partie III : Langage de définition de données	6
5 - En se basant sur le diagramme de classes fait, définir tous les types nécessaires. Prendre en compte toutes associations qui existent	6
6- Définir les tables nécessaires à la base de données	15
7- Définir les méthodes	21
A - Calculer pour chaque spécialité donnée, le nombre de médecins affectés	21
B- calculer pour chaque service donné, le nombre d'infirmier(ères) affecté(es) et le nombre de patients hospitalisés	22
C - calculer pour chaque patient le nombre total de ses médecins soignants	25
D - afficher « vérification positive » si le salaire de l'infirmier est entre 10000 DA et 30000 DA et affiche « Vérification négative » sinon	27
Partie IV : Langage de manipulation de données	29
Insertions des Tables	29
-Insertion table médecin	29
-Insertion table service	29
-Insertion table Infirmier	30
-Insertion table chambre	30
-Insertion table patient	31
Mise à jour des tables	32
-Mettre à jour la table chef_serv	32
-Mettre à jour la table serv_inf	32
-Mettre à jour la table serv_cham	33
-Mettre à jour la table inf_cham	33
-Mettre à jour la table cham_hospt	33
-Mettre à jour la table pat_med	34
-Mettre à jour la table med_pat	34
PARTIE V : Langage d'interrogation de données	36
9 - Donner la liste des patients (Prénom et nom) affiliés à la mutuelle « MAAF »	36
10 - Donner pour chaque lit occupé du bâtiment « B » de l'hôpital occupé par un patient affilié à une mutuelle dont le nom commence par « MN... », le numéro du lit, le numéro de la chambre, le nom du service ainsi que le prénom, le nom et la mutuelle du patient l'occupant	37
11 - Pour chaque patient soigné par plus de 3 médecins donner le nombre total de ses médecins ainsi que le nombre correspondant de spécialités médicales concernées	39
12 - Quelle est la moyenne des salaires des infirmiers(ères) par service ?	41
13 - Pour chaque service quel est le rapport entre le nombre d'infirmier(ères) affecté(es) au service et le nombre de patients hospitalisés dans le service ?	42
14 - Donner la liste des médecins (Prénom et nom) ayant un patient hospitalisé dans chaque service	43

PARTIE I : Modélisation orientée objet

1 - Transformez ce schéma relationnel en schéma Objet (diagramme de classes)

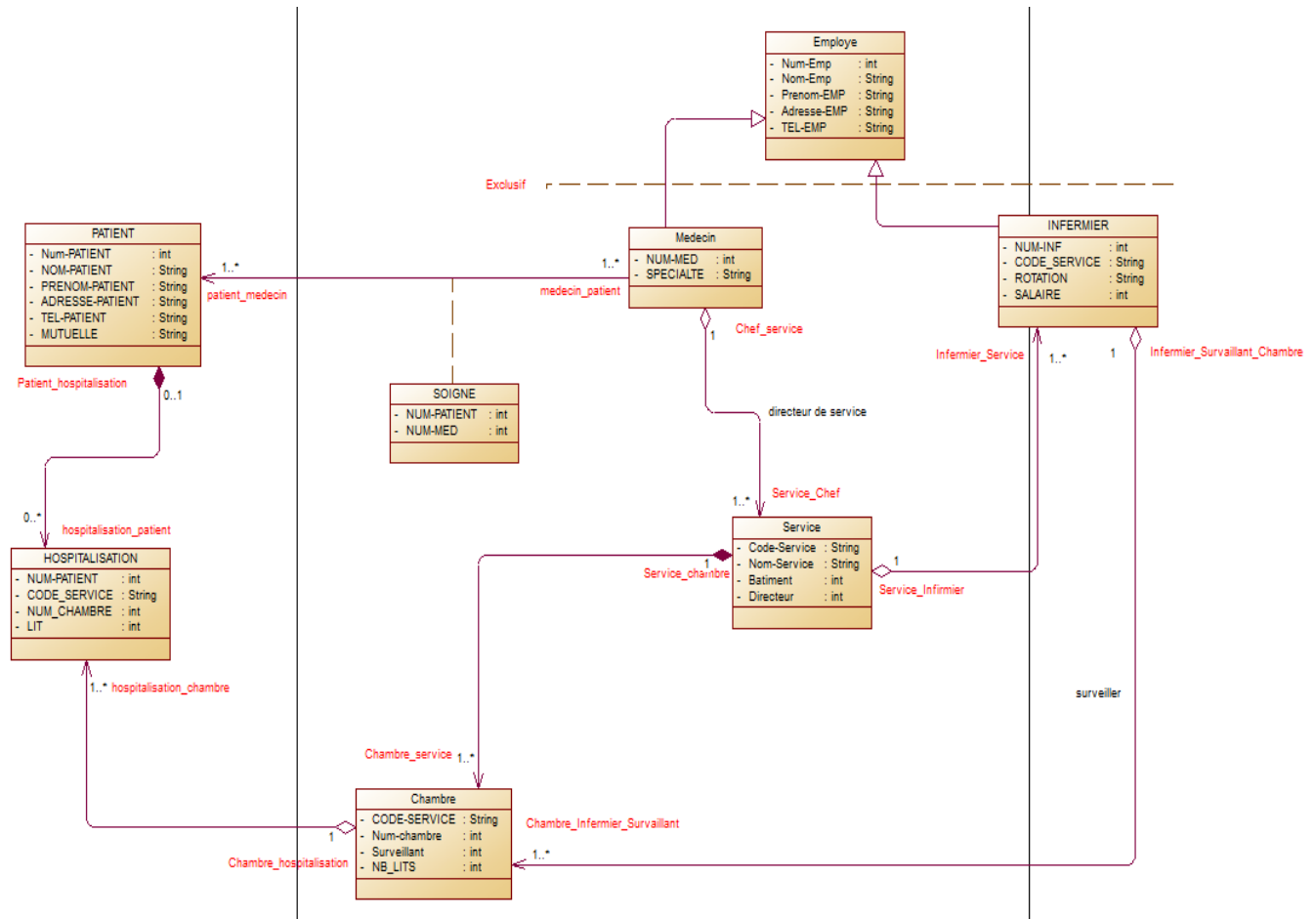


Figure 1 : Diagramme de classes

PARTIE II : Création des TableSpaces et utilisateur

2 - Créer deux TableSpaces SQL3 TBS et SQL3 TempTBS

```
create tablespace SQL3_TBS datafile 'C:\Users\HP\Documents\PYTHON CODECADEMY\TP S2\TP\TPBDA\SQL3_TBS.dat' size 100M autoextend on online;
```

```
SQL> create tablespace SQL3_TBS datafile 'C:\Users\HP\Documents\PYTHON CODECADEMY\TP S2\TP\TPBDA\SQL3_TBS.dat' size 100M autoextend on online;  
Tablespace created.
```

Figure 2.1 : Création de tableSpace sql3_tbs

```
create temporary tablespace SQL3_TempTBS tempfile 'C:\Users\HP\Documents\PYTHON CODECADEMY\TP S2\TP\TPBDA\SQL3_TempBS.dat' size 100M autoextend on;
```

```
SQL> create temporary tablespace SQL3_TempTBS tempfile 'C:\Users\HP\Documents\PYTHON CODECADEMY\TP S2\TP\TPBDA\SQL3_TempBS.dat' size 100M autoextend on;  
Tablespace created.
```

Figure 2.2 : Création de tableSpace sql3 temptbs

3 - Créer un utilisateur SQL3 en lui attribuant les deux tablespaces créés précédemment

```
create user DBAHOPITAL identified by 123 default tablespace SQL3_TBS temporary tablespace SQL3_TEMPPTS;
```

```
SQL> create user DBAHOPITAL identified by 123 default tablespace SQL3_TBS temporary tablespace SQL3_TempTBS;  
User created.
```

Figure 2.3 : Création du User DBAHOPITAL

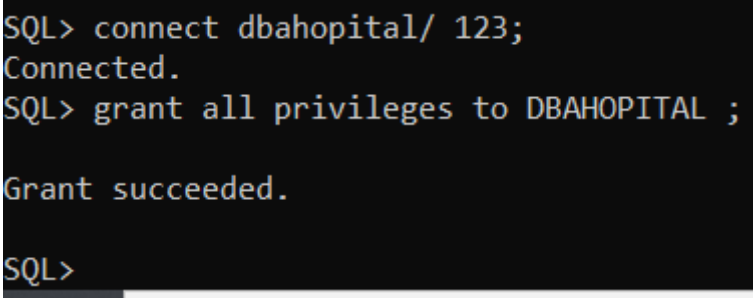
```
select username, created from dba_users where username=upper('dbahopital');
```

```
SQL> select username, created from dba_users where username=upper('dbahopital');  
  
USERNAME                                CREATED  
-----  
DBAHOPITAL                              16-APR-22
```

Figure 2.4 : Vérification de la création

4 – Donner tous les privilèges à cet utilisateur.

```
connect dbahopital/123;  
grant all privileges to DBAHOPITAL;
```

A screenshot of a terminal window with a black background and white text. It shows the execution of two SQL commands. The first command is 'SQL> connect dbahopital/ 123;', followed by the response 'Connected.'. The second command is 'SQL> grant all privileges to DBAHOPITAL ;', followed by the response 'Grant succeeded.'. The prompt 'SQL>' is visible at the bottom of the screenshot.

```
SQL> connect dbahopital/ 123;  
Connected.  
SQL> grant all privileges to DBAHOPITAL ;  
  
Grant succeeded.  
  
SQL>
```

Figure 2.5 : Donner tous les privilèges a dbaHopital

Partie III : Langage de définition de données

5 - En se basant sur le diagramme de classes fait, définir tous les types nécessaires. Prendre en compte toutes associations qui existent.

A - Définir tous les types nécessaires

Les types nécessaires sont les classes de notre diagramme les suivantes :

- Infirmier.
- Employe.
- Médecin.
- Service.
- Hospitalisation.
- Chambre.
- Patient.

On enlèvera la classe soigne car elle se compose des clés primaires des classe médecin et patient cette table sera remplacée par une table imbriquées entre les deux classes.

- Création des types incomplets

```
connect dbahopital/123;
```

```
SQL> connect dbahopital/ 123;  
Connected.
```

Figure 3.1 : Connection a notre user

Type employe

```
CREATE OR REPLACE TYPE temploye;  
/
```

```
SQL> CREATE OR REPLACE TYPE temploye;  
2 /  
  
Type created.
```

Figure 3.2 : Création du type temploye

Type medecin

```
CREATE OR REPLACE TYPE tmedecin;  
/
```

```
SQL> CREATE OR REPLACE TYPE tmedecin;  
2 /  
  
Type created.
```

Figure 3.3 : Création du type tmedecin

Type infirmier

```
CREATE OR REPLACE TYPE tinfirmier;  
/
```

```
SQL> CREATE OR REPLACE TYPE tinfirmier;  
2 /  
  
Type created.
```

Figure 3.4 : Création du type tinfirmier

Type Service

```
CREATE OR REPLACE TYPE tservice;  
/
```

```
SQL> CREATE OR REPLACE TYPE tservice;  
2 /  
  
Type created.
```

Figure 3.5 : Création du type tservice

Type chambre

```
CREATE OR REPLACE TYPE tchambre;  
/
```

```
SQL> CREATE OR REPLACE TYPE tchambre;  
2 /  
  
Type created.
```

Figure 3.6 : Création du type tchambre

Type patient

```
CREATE OR REPLACE TYPE tpatient;  
/
```

```
SQL> CREATE OR REPLACE TYPE tpatient;  
2 /  
  
Type created.
```

Figure 3.7 : Création du type tpatient

Type hospitalisation

```
CREATE OR REPLACE TYPE thospitalisation;  
/
```

```
SQL> CREATE OR REPLACE TYPE thospitalisation;  
2 /  
  
Type created.
```

Figure 3.8 : Création du type thospitalisation

B - Les associations

Créer les types nécessaires aux associations "les tables imbriquées des références"

D'après notre diagramme de classe nous allons créer des tables imbriquées pour références les classes ou leurs instances participe plusieurs fois.

Type tset_ref_medecin

```
CREATE OR REPLACE TYPE tset_ref_medecin AS TABLE OF REF tmedecin;  
/
```

```
SQL> CREATE OR REPLACE TYPE tset_ref_medecin as table of ref tmedecin;  
2 /  
  
Type created.
```

Figure 3.9 : Création du type tset_ref_medecin

Type tset_ref_infirmier

```
CREATE OR REPLACE TYPE tset_ref_infirmier AS TABLE OF REF tinfirmier;  
/
```

```
SQL> create or replace type tset_ref_infirmier as table of ref tinfirmier;  
2 /  
  
Type created.
```

Figure 3.10 : Création du type tset_ref_infirmier

Type tset_ref_patient

```
CREATE OR REPLACE TYPE tset_ref_patient AS TABLE OF REF tpatient;  
/
```

```
SQL> create or replace type tset_ref_patient as table of ref tpatient;  
2 /  
  
Type created.
```

Figure 3.11 : Création du type tset_ref_patient

Type tset_ref_service

```
CREATE OR REPLACE TYPE tset_ref_service AS TABLE OF REF tservice;
/

SQL> create or replace type tset_ref_service as table of ref tservice;
2 /

Type created.
```

Figure 3.12 : Création du type tset_ref_service

Type tset_ref_chambre

```
CREATE OR REPLACE TYPE tset_ref_chambre AS TABLE OF REF tchambre;
/

SQL> create or replace type tset_ref_chambre as table of ref tchambre;
2 /

Type created.
```

Figure 3.13 : Création du type tset_ref_chambre

C- Création des types

Nous utilisons notre modélisation pour créer nos types et attribuer les rôles.

Type employe

```
create or replace type temploye as object (
    num_emp INTEGER,
    nom_emp varchar2(50),
    prenom_emp varchar2(50),
    adresse_emp varchar2(100),
    tel_emp varchar2(100)
) not final;
/

SQL> create or replace type temploye as object (
2  num_emp INTEGER,
3  nom_emp varchar2(50),
4  prenom_emp varchar2(50),
5  adresse_emp varchar2(100),
6  tel_emp varchar2(100)
7  ) not final;
8  /

Type created.
```

Figure 3.14 : Création du type complet temploye

Type médecin

- Rôles

-**Medecin_patient** -> collection de références des objets de type tpatient.

-**Chef_service** -> collection de références des objets de type tservice.

-Nous avons utilisé les abréviations **med_pat**, **chef_serv** pour décrire les rôles **Medecin_patient** et **Chef_service**

```
create or replace type tmedecin under temploie (  
  specialite varchar2(30),  
  med_pat tset_ref_patient,  
  chef_serv tset_ref_service  
);  
/
```

```
SQL> create or replace type tmedecin under temploie (  
2  specialite varchar2(30),  
3  med_pat tset_ref_patient,  
4  chef_serv tset_ref_service  
5  );  
6  /
```

Type created.

Figure 3.15 : Création du type tmedecin

Type infirmier

-Rôles

-**Infirmier_service** -> sauvegarde la référence du service attribués à cet infirmier.

-**Infirmier_surveillant_chambre** -> collection de références des objets de type tchambre.

-Nous avons utilisé les abréviations **inf_serv**, **Inf_surveillant_cham** pour décrire les rôles **Infirmier_service** et **Infirmier_surveillant_chambre**.

```
create or replace type tinfirmier under temploie(  
  rotation varchar2(4),  
  salaire float(2),  
  inf_serv ref tservice,  
  inf_surveillant_cham tset_ref_chambre  
);  
/
```

```
SQL> create or replace type tinfirmier under temploie(  
  2 rotation varchar2(4),  
  3 salaire float(2),  
  4 inf_serv ref tservice,  
  5 inf_surveillant_cham tset_ref_chambre  
  6 );  
  7 /
```

Type created.

Figure 3.16 : Création du type complet tinfirmier

Type service

-Rôles

-**service_chef** -> sauvegarde la référence du directeur concerné par ce service.

-**service_Infirmier** -> collection de références des objets de type tinfirmer.

-**service_chambre** -> collection de références des objets de type tchambre.

-Nous avons utilisé les abréviations **serv_chef**, **serv_Inf**, **serv_cham** pour décrire les rôles **service_chef**, **service_Infirmier** et **service_chambre**.

```
create or replace type tservice as object (  
  code_service varchar2(3),  
  nom_service varchar2(50),  
  batiment varchar2(1),  
  serv_chef ref tmedecin,  
  serv_inf tset_ref_infirmier,  
  serv_cham tset_ref_chambre  
);  
/
```

```
SQL> create or replace type tservice as object (  
  2  code_service varchar2(3),  
  3  nom_service varchar2(50),  
  4  batiment varchar2(1),  
  5  serv_chef ref tmedecin,  
  6  serv_inf tset_ref_infirmier,  
  7  serv_cham tset_ref_chambre  
  8  );  
  9  /  
  
Type created.
```

Figure 3.17 : Création du type complet tservice

Type chambre

- **chambre_service** -> sauvegarde la référence du service concerné par cette chambre.
 - **chambre_hospitalisation** -> collection de références des objets de type tpatient.
 - **chambre_infirmier_surveillant** -> sauvegarde la référence du infirmier concerné par cette chambre.
- Nous avons utilisé les abréviations **serv_chef**, **serv_Inf**, **cham_inf_surveillant** pour décrire les rôles **chambre_service**, **chambre_hospitalisation** et **chambre_infirmier_surveillant**.

```
create or replace type tchambre as object (  
  code_service VARCHAR2(3),  
  num_chambre INTEGER,  
  nb_lits INTEGER,  
  cham_serv REF tservice,  
  cham_hospt tset_ref_patient,  
  cham_inf_surveillant REF tinfirmier  
);  
/
```

```
SQL> create or replace type tchambre as object (  
  2 code_service varchar2(3),  
  3 num_chambre INTEGER,  
  4 nb_lits INTEGER,  
  5 cham_serv ref tservice,  
  6 cham_hospt tset_ref_patient,  
  7 cham_inf_surveillant ref tinfirmier  
  8 );  
  9 /  
  
Type created.
```

Figure 3.18 : Création du type complet tchambre

Type patient

-Rôles

- **patient_medecin** -> collection de références des objets de type tmedecin.
- **Hospt** -> sauvegarde la type structure du type hospitalisation.
- Nous avons utilisé les abréviations **pat_med** pour décrire les rôles **patient_medecin**.

```
create or replace type tpatient as object (  
  num_patient INTEGER,  
  nom_patient varchar2(50),  
  prenom_patient varchar2(50),  
  adresse_patient varchar2(100),  
  tel_patient varchar2(12),  
  mutuelle varchar2(10),  
  pat_med tset_ref_medecin,  
  hospt thospitalisation  
);  
/
```

```
SQL> create or replace type tpatient as object (  
 2  num_patient INTEGER,  
 3  nom_patient varchar2(50),  
 4  prenom_patient varchar2(50),  
 5  adresse_patient varchar2(100),  
 6  tel_patient varchar2(12),  
 7  mutuelle varchar2(10),  
 8  pat_med tset_ref_medecin,  
 9  hospt thospitalisation  
10 );  
11 /  
  
Type created.
```

Figure 3.19 : Création du type complet tpatient

Type hospitalisation

-Rôles

- **hospitalisation_chambre** -> sauvegarde la référence de la chambre concernée par cette hospitalisation.

-Nous avons utilisé les abréviations **hospt_cham** pour décrire les rôles **hospitalisation_chambre**.

```
create or replace type thospitalisation as object (  
  lit INTEGER,  
  hospt_cham ref tchambre  
);  
/
```

```
SQL> create or replace type thospitalisation as object (  
  2  lit INTEGER,  
  3  hospt_cham ref tchambre  
  4  );  
  5  /  
  
Type created.
```

Figure 3.20 : Création du type complet thospitalisation

6- Définir les tables nécessaires à la base de données.

Les tables choisies à être utiliser sont :

-Table medecin.

-Table infirmier.

-Table sservice.

-Table chambre.

- Table patient.

- Nous avons choisies d'utiliser la table médecin et infirmier a la place de la table employe et cela pour nous faciliter le travail lors de la manipulation de donnees.

-Nous avons nommer la table service en sservice pour avoir aucun problème dans le cas ou le mot soit réservés.

Table médecin

-La clé primaire est num_emp.

-On codifie la collection med_pat en table_med_pat.

-On codifie la collection chef_serv en table_chef_serv.

```
create table medecin of tmedecin (  
  constraint pk_med primary key(num_emp))  
  nested table med_pat store as table_med_pat,  
  nested table chef_serv store as table_chef_serv;
```

```
SQL> create table medecin of tmedecin (  
  2  constraint pk_med primary key(num_emp))  
  3  nested table med_pat store as table_med_pat,  
  4  nested table chef_serv store as table_chef_serv;  
  
Table created.
```

Figure 3.21 : Création de la table medecin

Verification

```
desc medecin;
```

```
SQL> desc medecin  
Name                               Null?      Type  
-----  
NUM_EMP                            NOT NULL   NUMBER(38)  
NOM_EMP                            VCHAR2(50)  
PRENOM_EMP                         VCHAR2(50)  
ADRESSE_EMP                        VCHAR2(100)  
TEL_EMP                            VCHAR2(100)  
SPECIALITE                        VCHAR2(30)  
MED_PAT                           TSET_REF_PATIENT  
CHEF_SERV                         TSET_REF_SERVICE
```

Figure 3.22 : Verification de la table medecin

Table service

- La clé primaire est code_service.
- La clé étrangère est serv_chef.
- On codifie la collection serv_inf en table_serv_inf.
- On codifie la collection serv_cham en table_serv_cham.

```
create table sservice of tservice (  
  constraint pk_serv primary key(code_service),  
  constraint fk_serv foreign key (serv_chef) references medecin)  
  nested table serv_inf store as table_serv_inf,  
  nested table serv_cham store as table_serv_cham;
```

```
SQL> create table sservice of tservice (  
  2  constraint pk_serv primary key(code_service),  
  3  constraint fk_serv foreign key (serv_chef) references medecin)  
  4  nested table serv_inf store as table_serv_inf,  
  5  nested table serv_cham store as table_serv_cham;
```

Table created.

Figure 3.22 : Création de la table service

Verification

```
desc sservice;
```

```
SQL> desc sservice
```

Name	Null?	Type
CODE_SERVICE	NOT NULL	VARCHAR2(3)
NOM_SERVICE		VARCHAR2(50)
BATIMENT		VARCHAR2(1)
SERV_CHEF		REF OF TMEDECIN
SERV_INF		TSET_REF_INFERMIER
SERV_CHAM		TSET_REF_CHAMBRE

Figure 3.23 : Verification de la table service

Table infirmier

- La clé primaire est num_emp.
- La clé étrangère est inf_serv.
- On fait le check sur la rotation.
- On codifie la collection inf_surveillant_cham en table_inf_surveillant_cham.

```
create table infirmier of tinfirmier (  
  constraint pk_inf primary key(num_emp),  
  constraint fk_inf foreign key(inf_serv) references sservice,  
  check (rotation in ('NUIT','JOUR'))  
  nested table inf_surveillant_cham store as table_inf_surveillant_cham;  
SQL> create table infirmier of tinfirmier (  
  2  constraint pk_inf primary key(num_emp),  
  3  constraint fk_inf foreign key(inf_serv) references sservice,  
  4  check (rotation in ('NUIT','JOUR'))  
  5  nested table inf_surveillant_cham store as table_inf_surveillant_cham;  
Table created.
```

Figure 3.24 : Création de la table infirmier

Vérification

```
desc infirmier;  
SQL> desc infirmier  
Name                               Null?      Type  
-----  
NUM_EMP                            NOT NULL  NUMBER(38)  
NOM_EMP                            VCHAR2(50)  
PRENOM_EMP                         VCHAR2(50)  
ADRESSE_EMP                        VCHAR2(100)  
TEL_EMP                            VCHAR2(100)  
ROTATION                           VCHAR2(4)  
SALAIRE                            FLOAT(2)  
INF_SERV                           REF OF TSERVICE  
INF_SURVEILLANT_CHAM               TSET_REF_CHAMBRE
```

Figure 3.25 : Vérification de la table infirmier

Table chambre

- La clé primaire est (code_service, num_chambre).
- La clé étrangère est cham_serv.
- La clé étrangère est cham_inf_surveillant.
- On fait le check si le nombre de lit est > 0.
- On codifie la collection cham_hospt en table_cham_hospt.

```
create table chambre of tchambre (
constraint pk_cham primary key(code_service,num_chambre),
constraint fk_cham_inf foreign key(cham_inf_surveillant) references infirmier,
constraint fk_cham_serv foreign key(cham_serv) references sservice)
nested table cham_hospt store as table_cham_hospt;
```

```
SQL> create table chambre of tchambre (
2 constraint pk_cham primary key(code_service,num_chambre),
3 constraint fk_cham_inf foreign key(cham_inf_surveillant) references infirmier,
4 constraint fk_cham_serv foreign key(cham_serv) references sservice)
5 nested table cham_hospt store as table_cham_hospt;
```

Table created.

Figure 3.26 : Création de la table chambre

```
alter table chambre add constraint ck_lit check (lit>0);
```

```
SQL> alter table chambre add constraint ck_lit check (nb_lits>0);
Table altered.
```

Figure 3.27 : Ajout de la contrainte sur le lit

Verification

```
desc chambre;
```

```
SQL> desc chambre
```

Name	Null?	Type
CODE_SERVICE	NOT NULL	VARCHAR2(3)
NUM_CHAMBRE	NOT NULL	NUMBER(38)
NB_LITS		NUMBER(38)
CHAM_SERV		REF OF TSERVICE
CHAM_HOSPT		TSET_REF_PATIENT
CHAM_INF_SURVAILLANT		REF OF TINFERMIER

Figure 3.28 : Verification de la table chambre

Table patient

- La clé primaire est num_patient.
- La clé étrangère est hospt.hospt_cham.
- On codifie la collection pat_med en table_pat_med.

```
create table patient of tpatient (  
  constraint pk_pat primary key(num_patient),  
  constraint fk_pat foreign key(hospt.hospt_cham) references chambre)  
  nested table pat_med store as table_pat_med;
```

```
SQL> create table patient of tpatient (  
  2  constraint pk_pat primary key(num_patient),  
  3  constraint fk_pat foreign key(hospt.hospt_cham) references chambre)  
  4  nested table pat_med store as table_pat_med;
```

Table created.

Figure 3.29 : Création de la table patient

Verification

```
desc patient;
```

```
SQL> desc patient
```

Name	Null?	Type
NUM_PATIENT	NOT NULL	NUMBER(38)
NOM_PATIENT		VARCHAR2(50)
PRENOM_PATIENT		VARCHAR2(50)
ADRESSE_PATIENT		VARCHAR2(100)
TEL_PATIENT		VARCHAR2(12)
MUTUELLE		VARCHAR2(10)
PAT_MED		TSET_REF_MEDECIN
HOSPT		THOSPITALISATION

Figure 3.30 : Verification de la table patient

7- Définir les méthodes

A - Calculer pour chaque spécialité donnée, le nombre de médecins affectés.

```
SET SERVEROUTPUT ON  
alter type tmedecin add static function nb_spe(spec varchar2) return INTEGER cascade;  
SQL> SET SERVEROUTPUT ON  
SQL> alter type tmedecin add static function nb_spe(spec varchar2) return INTEGER cascade;  
Type altered.
```

Figure 3.31 : Ajout de la methode nb_spe

```
create or replace type BODY tmedecin as static function nb_spe(spec varchar2)  
return integer is nb_med integer;  
Begin  
select count(med.num_emp) into nb_med  
from medecin med  
where med.specialite = spec  
group by med.specialite;  
return nb_med;  
end;  
end;  
/
```

Explication

Dans cette méthode de classe nous utilisons une fonction sur le type tmedecin.

Nous utilisons la table médecin seulement. Nous comptons le nombre de médecin pour chaque spécialité puis nous les regroupons selon chacune d'elle, nous retournons ensuite le resultat dans nb_med.

```
SQL> create or replace type BODY tmedecin as static function nb_spe(spec varchar2)  
2 return integer is nb_med integer;  
3 Begin  
4 select count(med.num_emp) into nb_med  
5 from medecin med  
6 where med.specialite = spec  
7 group by med.specialite;  
8 return nb_med;  
9 end;  
10 end;  
11 /  
Type body created.
```

Figure 3.32 : Creation de la methode nb_spe

Vérification après insertion des tuples

```
select distinct specialite, tmedecin.nb_spe(specialite) from medecin;  
SQL> select distinct specialite, tmedecin.nb_spe(specialite) from medecin;
```

Figure 3.32 : Requete de verification de la methode nb_spe

Le Résultat

SPECIALITE	TMEDECIN.NB_SPE(SPECIALITE)
Anesthésiste	5
Pneumologue	5
Cardiologue	7
Traumatologue	5
Radiologue	4
Cardiologue	1
Orthopédiste	5

7 rows selected.

Figure 3.33 : Resultat de la requete de la methode nb_spe

Analyse des résultats

Nous remarquons qu'il y a un nombre stable de médecin dans chaque spécialité donc une bonne répartition avec un léger nombre de cardiologue en plus par rapport au reste.

B- calculer pour chaque service donné, le nombre d'infirmier(ères) affecté(es) et le nombre de patients hospitalisés.

```
alter type tservice add member function nb_pat return INTEGER cascade;
```

```
SQL> alter type tservice add member function nb_pat return INTEGER cascade;  
Type altered.
```

Figure 3.34 : Ajout de la methode nb_pat

```
alter type tservice add member function nb_inf return INTEGER cascade;  
SQL> alter type tservice add member function nb_inf return INTEGER cascade;  
Type altered.
```

Figure 3.35 : Ajout de la methode nb_inf

```

create or replace type BODY tservice as member function nb_inf
return integer is result integer;
Begin
select count(Distinct serInf.column_value) into result
from table(self.serv_inf) serInf;
return result;
end nb_inf;
member function nb_pat return integer is result1 integer;
Begin
select count(distinct chamHospt.column_value) into result1
from table(self.serv_cham) servCham, chambre cham, table(cham.cham_hospt) chamHospt
where servCham.column_value = REF(cham);
return result1;
end nb_pat;
end;
/

```

Explication

Dans cette méthode d'instance nous utilisons une fonction sur le type tservice.

Pour la fonction nb_inf nous utilisons la table imbriquée serv_inf on utilise la fonction value sur cette table pour retourner l'instance de type tinfirmier et on compte la valeur distincte de chaque tuples puis nous retournons le résultat.

Pour la fonction nb_pat, nous utilisons la table chambre et les chambres imbriquées serv_cham et cham_hospt. On utilise la fonction value sur la table cham_hospt qui nous retourne l'instance du type tpatient et on compte la valeur distincte de chaque tuple ou l'instance retourner par la fonction value sur le serv_cham soit égale à la référence de la table cham de la chambre pour retourner le résultat.

```

SQL> create or replace type BODY tservice as member function nb_inf
  2  return integer is result integer;
  3  Begin
  4  select count(Distinct serInf.column_value) into result
  5  from table(self.serv_inf) serInf;
  6  return result;
  7  end nb_inf;
  8  member function nb_pat return integer is result1 integer;
  9  Begin
 10  select count(distinct chamHospt.column_value) into result1
 11  from table(self.serv_cham) servCham, chambre cham, table(cham.cham_hospt) chamHospt
 12  where servCham.column_value = REF(cham);
 13  return result1;
 14  end nb_pat;
 15  end;
 16  /

Type body created.

```

Figure 3.36 : Creation de la methode nb_inf et nb_pat

Vérification après insertion des tuples

Nb infirmier

```
SELECT serv.code_service, serv.nb_inf() AS "nb infirmier"  
FROM sservice serv;
```

```
SQL> SELECT serv.code_service, serv.nb_inf() AS "nb infirmier"  
2  from sservice serv;
```

Figure 3.37 : Requete de verification de la methode nb_inf

Le résultat

COD	nb infirmier
CAR	8
REA	7
CHG	13

Figure 3.38 : Resultat de la requete sur de la methode nb_inf

Nb Patient

```
SELECT serv.code_service, serv.nb_pat() AS "nb patient"  
FROM sservice serv;
```

```
SQL> SELECT serv.code_service, serv.nb_pat() AS "nb patient"  
2  from sservice serv;
```

Figure 3.39 : Requete de verification de la methode nb_pat

Le résultat

COD	nb patient
CAR	9
REA	10
CHG	20

Figure 3.40 : Resultat de la requete sur de la methode nb_pat

Analyse des résultats

Nous remarquons que le nombre de patients et d'infirmiers le plus élevés se trouve au niveau du service chirurgie générales. Ce qui est logique de trouver bien plus d'infirmiers là où il y a plus de malade. On a un équilibre dans le nombre de patients et d'infirmiers pour les services cardiologie et réanimation.

C - calculer pour chaque patient le nombre total de ses médecins soignants.

```
alter type tpatient add member function nb_medecin return integer cascade;
SQL> alter type tpatient add member function nb_medecin return integer cascade;
Type altered.
```

Figure 3.41 : Creation de la methode nb_medecin

```
create or replace type BODY tpatient as member function nb_medecin
return integer is nbMedResult integer;
Begin
select count(distinct pat.column_value) into nbMedResult
from table(self.pat_med) pat;
return nbMedResult;
end nb_medecin;
end;
/
```

Explication

Dans cette méthode d'instance nous utilisons une fonction sur le type tpatient.

Le résultat retourner par la fonction sera retourner dans la variable NBMEDRESULT. Nous utilisons la table imbriquée pat_med. On utilise la fonction value sur cette table pour retourner l'instance de type tmedecin et on compte la valeur distincte de chaque tuples de la table médecin puis nous retournons le résultat.

```
SQL> create or replace type BODY tpatient as member function nb_medecin
2  return integer is nbMedResult integer;
3  Begin
4  select count(distinct pat.column_value) into nbMedResult
5  from table(self.pat_med) pat;
6  return nbMedResult;
7  end nb_medecin;
8  end;
9  /
Type body created.
```

Figure 3.42 : Ajout de la methode nb_medecin

Vérification des résultats

```
SELECT pat.num_patient, pat.nb_medecin() AS "nb medecin"
FROM patient pat;
```

```
SQL> SELECT pat.num_patient, pat.nb_medecin() AS "nb medecin"
2 FROM patient pat;
```

Figure 3.43 : requete de verification de la methode nb_medecin

Le résultat

NUM_PATIENT	nb medecin	NUM_PATIENT	nb medecin
1	2	81	2
3	3	90	1
6	4	91	2
13	3	92	1
14	1	100	2
21	3	101	1
23	3	102	2
33	2	103	3
35	2	104	1
36	3	105	3
37	1	107	2
41	3	108	4
43	2	109	1
44	2	117	4
46	2	119	3
52	3	120	2
55	2	121	1
56	2	123	3
60	1	124	1
61	3	128	2
63	2	133	1
65	1	137	2
66	1	138	1
67	1	145	4
68	3	146	2
70	2	147	4
72	1	148	1
74	2	149	1
75	1	153	2
76	4	154	2
77	2	158	1
78	2	159	4
79	1	164	2
166	1	167	1
168	1	171	2
172	4	175	1
181	2	182	3
184	3	187	1
188	1	190	1
191	1	192	3

Figure 3.44 : Resultat de la requete sur de la methode nb_medecin

Analyse des résultats

- Nous remarquons que le nombre de médecin soignants pour chaque patient est plutôt stable, nous trouvons qu'il visite deux médecins pour la plus part. Ceci est plutôt bon pour augmenter le chiffre d'affaires de l'hôpital et qui montre que les médecins sont compétents dans leurs travaux.

D - afficher « vérification positive » si le salaire de l'infirmier est entre 10000 DA et 30000 DA et affiche « Vérification négative » sinon.

```
alter type tinfirmier add member procedure verif_sal cascade;

SQL> alter type tinfirmier add member procedure verif_sal cascade;

Type altered.
```

Figure 3.45 : Creation de la methode verif_sal

```
create or replace type BODY tinfirmier as member procedure verif_sal is
BEGIN if self.salaire BETWEEN 10000 and 30000 then
  DBMS_output.put_line('verification positive');
else DBMS_output.put_line('verification negative');
end if;
end verif_sal;
end;
/
```

-Explication

Dans cette méthode d'instance nous utilisons une procédure sur le type tinfirmier.

Nous utilisons la table infirmier seulement nous utilisons le self.salaire sur la table infirmier pour voir si son salaire est entre 10000 et 30000, on retourne le message « vérification positive » s'il est vrai ou « vérification négative » dans le cas contraire.

```
SQL> create or replace type BODY tinfirmier as member procedure verif_sal is
 2 BEGIN if self.salaire BETWEEN 10000 and 30000 then
 3   DBMS_output.put_line('verification positive');
 4 else DBMS_output.put_line('verification negative');
 5 end if;
 6 end verif_sal;
 7 end;
 8 /

Type body created.
```

Figure 3.46 : Creation de la methode verif_sal

Vérification des résultats

```
DECLARE
  inf TINFERMIER;
BEGIN
  SELECT value(i) INTO inf
  FROM INFERMIER i
  WHERE i.num_emp = 12;
  inf.verif_sal;
END;
/
```

```
SQL> SET SERVEROUTPUT ON
SQL> DECLARE
2      inf TINFERMIER;
3  BEGIN
4      SELECT value(i) INTO inf
5      FROM INFERMIER i
6      WHERE i.num_emp = 12;
7      inf.verif_sal;
8  END;
9  /
verification positive

PL/SQL procedure successfully completed.
```

Figure 3.47 : Resultat de la requete sur de la methode verif_sal

Analyse des résultats

Nous remarquons que tous les infirmiers dans l'hôpital ont une vérification positive. On conclue que les infirmiers sont payer à leurs juste valeur et qu'il n'y a aucune discrimination.

Partie IV : Langage de manipulation de données

Insertions des Tables

-Insertion table médecin

```
INSERT INTO medecin VALUES(tmedecin(4,'BOUROUBI','Taous','Lotissement Dauphin n°30  
DRARIA/ALGER','021356085','Orthopédiste',tset_ref_patient(),tset_ref_service()));  
SQL> INSERT INTO medecin VALUES(tmedecin(4,'BOUROUBI','Taous','Lotissement Dauphin n°30 DRARIA/ALGER',  
2 '021356085','Orthopédiste',tset_ref_patient(),tset_ref_service()));  
1 row created.
```

Figure 4.1 : Insertion du tuple dans la table medecin

Verification

```
select count(*) from medecin;  
SQL> select count(*) from medecin;  
  
COUNT(*)  
-----  
32
```

Figure 4.2 : Nombre de tuple dans la table medecin

-Insertion table sservice

```
INSERT INTO SSERVICE VALUES(tservice('REA','Réanimation et Traumatologie','A',  
(select ref(med) from medecin med where  
num_emp=19),tset_ref_infirmier(),tset_ref_chambre()));  
SQL> INSERT INTO SSERVICE VALUES(tservice('REA','Réanimation et Traumatologie','A',  
2 (select ref(med) from medecin med where num_emp=19),tset_ref_infirmier(),tset_ref_chambre()));  
1 row created.
```

Figure 4.3 : Insertion du tuple dans la table sservice

Verification

```
select count(*) from sservice;  
SQL> select count(*) from sservice;  
  
COUNT(*)  
-----  
3
```

Figure 4.4 : Nombre de tuple dans la table sservice

-Insertion table Infirmier

```
INSERT INTO INFIRMIER VALUES(tinfirmier(12,'HADJ','Zouhir','Cité de la Mosquée Bt 14-  
Boufarik-Blida',  
'025474882','JOUR',12560.78,(select ref(serv) FROM  
sservice serv where code_service='REA'),tset_ref_chambre()));
```

```
SQL> INSERT INTO INFIRMIER VALUES(tinfirmier(22,'ABAD',  
2 'Abdelhamid','8 Cours Aissat Idir-El Harrach-Alger','021524587',  
3 'JOUR',14980.21,(select ref(serv) FROM  
4 sservice serv where code_service='REA'),tset_ref_chambre()));  
  
1 row created.
```

Figure 4.5 : Insertion du tuple dans la table infirmier

Verification

```
select count(*) from infirmier;
```

```
SQL> select count(*) from infirmier;  
  
COUNT(*)  
-----  
28
```

Figure 4.6 : Nombre de tuple dans la table infirmier

-Insertion table chambre

```
INSERT INTO CHAMBRE VALUES (tchambre('CAR',101,3,(select ref(serv) FROM sservice serv  
where code_service='CAR'),  
tset_ref_patient(),(select ref(inf) FROM infirmier inf where num_emp=95)));
```

```
SQL> INSERT INTO CHAMBRE VALUES (tchambre('CAR',101,3,  
2 (select ref(serv) FROM sservice serv where code_service='CAR'),  
3 tset_ref_patient(),(select ref(inf) FROM infirmier inf where num_emp=95)));  
  
1 row created.
```

Figure 4.7 : Insertion du tuple dans la table chambre

Verification

```
select count(*) from chambre;
```

```
SQL> select count(*) from chambre;  
  
COUNT(*)  
-----  
24
```

Figure 4.8 : Nombre de tuple dans la table chambre

-Insertion table patient

-Sans-hospitalisation

```
INSERT INTO PATIENT VALUES(tpatient(13,'MAHBOUBA','Cherifa','CITE 1013 LOGTS BT 61  
KHROUB- Constantine','031966095','MAAF',tset_ref_medecin(),null));
```

```
SQL> INSERT INTO PATIENT VALUES(tpatient(13,'MAHBOUBA','Cherifa','CITE 1013 LOGTS BT 61 KHROUB- Constantine',  
2 '031966095','MAAF',tset_ref_medecin(),null));  
  
1 row created.
```

Figure 4.9 : Insertion du tuple sans hospitalisation dans la table patient

-Avec-hospitalisation

```
INSERT INTO PATIENT VALUES(tpatient(1,'GRIGAHCINE','Nacer','95,Bd Bougara-El biar-Alger',  
'021920313','MNAM',tset_ref_medecin(),THOSPITALISATION(1,  
(select ref(cham) from chambre cham where num_chambre=101 and code_service='REA'))));
```

```
SQL> INSERT INTO PATIENT VALUES(tpatient(1,'GRIGAHCINE','Nacer','95,Bd Bougara-El biar-Alger',  
2 '021920313','MNAM',tset_ref_medecin(),THOSPITALISATION(1,  
3 (select ref(cham) from chambre cham where num_chambre=101 and code_service='REA'))));  
  
1 row created.
```

Figure 4.10 : Insertion du tuple avec hospitalisation dans la table patient

Verification

```
select count(*) from patient;
```

```
SQL> select count(*) from patient;  
  
COUNT(*)  
-----  
80
```

Figure 4.11 : Nombre de tuple dans la table patient

Mise à jour des tables

-Mettre à jour la table chef_serv

```
insert into table (select med.chef_serv from medecin med where num_emp=80)
(select ref(serv) from sservice serv where code_service='CAR');
```

```
SQL> insert into table (select med.chef_serv from medecin med where num_emp=34)
  2  (select ref(serv) from sservice serv where code_service='CHG');
1 row created.
```

Figure 4.12 : Mise à jour de la table chef_serv

Verification

```
select count(*) from medecin med, table(med.chef_serv) ;
```

```
SQL> select count(*) from medecin med, table(med.chef_serv) ;

COUNT(*)
-----
        3
```

Figure 4.13 : Nombre de tuple de la table chef_serv

-Mettre à jour la table serv_inf

```
insert into table (select serv.serv_inf from sservice serv where code_service='REA')
(select ref(inf) from infermier inf where num_emp=12);
```

```
SQL> insert into table (select serv.serv_inf from sservice serv where code_service='REA')
  2  (select ref(inf) from infermier inf where num_emp=12)
  3  ;
1 row created.
```

Figure 4.14 : Mise à jour de la table serv_inf

Verification

```
select count(*) from sservice serv, table(serv.serv_inf) ;
```

```
SQL> select count(*) from sservice serv, table(serv.serv_inf) ;

COUNT(*)
-----
       28
```

Figure 4.15 : Nombre de tuple de la table serv_inf

-Mettre à jour la table serv_cham

```
insert into table (select serv.serv_cham from sservice serv where code_service='CAR')
(select ref(cham) from chambre cham where num_chambre=101 and code_service='CAR');
SQL> insert into table (select serv.serv_cham from sservice serv where code_service='CAR')
2 (select ref(cham) from chambre cham where num_chambre=101 and code_service='CAR');
1 row created.
```

Figure 4.16 : Mise à jour de la table serv_cham

Verification

```
select count(*) from sservice serv, table(serv.serv_cham) ;
SQL> select count(*) from sservice serv, table(serv.serv_cham) ;
COUNT(*)
-----
24
```

Figure 4.17 : Nombre de tuple de la table serv_cham

-Mettre à jour la table inf_cham

```
insert into table (select inf.inf_surveillant_cham from infirmier inf where num_emp=95)
(select ref(cham) from chambre cham where num_chambre=101 and code_service='CAR');
SQL> insert into table (select inf.inf_surveillant_cham from infirmier inf where num_emp=95)
2 (select ref(cham) from chambre cham where num_chambre=101 and code_service='CAR');
1 row created.
```

Figure 4.18 : Mise à jour de la table inf_cham

Verification

```
select count(*) from infirmier inf, table(inf.inf_surveillant_cham) ;
SQL> select count(*) from infirmier inf, table(inf.inf_surveillant_cham) ;
COUNT(*)
-----
24
```

Figure 4.19 : Nombre de tuple de la table inf_cham

-Mettre à jour la table cham_hospt

```
insert into table (select cham.cham_hospt from chambre cham where num_chambre=101 and
code_service='CAR')
(select ref(pat) from patient pat where num_patient=68);
SQL> insert into table (select cham.cham_hospt from chambre cham where num_chambre=101 and code_service='CAR')
2 (select ref(pat) from patient pat where num_patient=68);
1 row created.
```

Figure 4.20 : Mise à jour de la table cham_hospt

Verification

```
select count(*) from chambre cham, table(cham.cham_hospt);
```

```
SQL> select count(*) from chambre cham, table(cham.cham_hospt)
2  ;

COUNT(*)
-----
39
```

Figure 4.21 : Nombre de tuple de la table cham_hospt

-Mettre à jour la table pat_med

```
insert into table (select pat.pat_med from patient pat where num_patient=13)
(select ref(med) from medecin med where num_emp=4);
```

```
SQL> insert into table (select pat.pat_med from patient pat where num_patient=13)
2  (select ref(med) from medecin med where num_emp=4);

1 row created.
```

Figure 4.22 : Mise à jour de la table pat_med

Verification

```
select count(*) from patient pat, table(pat.pat_med) ;
```

```
SQL>
SQL> select count(*) from patient pat, table(pat.pat_med) ;

COUNT(*)
-----
163
```

Figure 4.23 : Nombre de tuple de la table pat_med

-Mettre à jour la table med_pat

```
insert into table (select med.med_pat from medecin med where num_emp=4)
(select ref(pat) from patient pat where num_patient=13);
```

```
SQL> insert into table (select med.med_pat from medecin med where num_emp=4)
2  (select ref(pat) from patient pat where num_patient=13);

1 row created.
```

Figure 4.24 : Mise à jour de la table med_pat

Verification

```
select count(*) from medecin med, table(med.med_pat) ;
```

```
SQL> select count(*) from medecin med, table(med.med_pat) ;  
  
COUNT(*)  
-----  
163
```

Figure 4.25 : Nombre de tuple de la table med_pat

PARTIE V : Langage d'interrogation de données

9 - Donner la liste des patients (Prénom et nom) affiliés à la mutuelle « MAAF ».

```
SELECT pat.nom_patient AS "nom" ,pat.prenom_patient AS "prenom"  
FROM patient pat  
WHERE pat.mutuelle = 'MAAF';
```

```
SQL> select pat.nom_patient AS "NOM",pat.prenom_patient AS "PRENOM"  
2  from patient pat  
3  where pat.mutuelle = 'MAAF';
```

Figure 5.1 : Requete liste de patient

Le résultat de la requête

nom	nom	nom
-----	-----	-----
prenom	prenom	prenom
-----	-----	-----
MAHBOUBA	LAAOUAR	MATI
Cherifa	Ali	Djamel
BOUDJELAL	MEDJAHED	HABABB
Salim	Ahmed	khadra
DIAF AMROUNI	HALFAOUI	
Ghania	Redouane	
		8 rows selected.

Figure 5.2 : Resultat de la requete liste de patient

Analyse des résultats

Il n'y a pas un grand nombre de patients de la mutuelle MAAF qui ont été soignés dans cet hôpital.

10 - Donner pour chaque lit occupé du bâtiment « B » de l'hôpital occupé par un patient affilié à une mutuelle dont le nom commence par « MN... », le numéro du lit, le numéro de la chambre, le nom du service ainsi que le prénom, le nom et la mutuelle du patient l'occupant.

```
SELECT pat.hospt.lit AS "numero lit",
       cham.num_chambre AS "numero chambre",
       Deref(cham.cham_serv).nom_service AS "nom service",
       pat.nom_patient AS "nom patient",
       pat.prenom_patient AS "prenom patient",
       pat.mutuelle AS "mutuelle"
FROM patient pat, chambre cham
WHERE pat.mutuelle LIKE 'MN%' and Deref(cham.cham_serv).batiment = 'B'
      and pat.hospt IS NOT NULL and
      Deref(pat.hospt.hospt_cham).code_service = cham.code_service and
      Deref(pat.hospt.hospt_cham).num_chambre = cham.num_chambre;
```

-Explication

Nous prenons la table patient et la table chambre de la table patient nous prenons le nom, le prénom et la mutuelle du patient. Nous utilisons l'objet pat.hospt pour prendre le numéro de lit. Nous utilisons la table chambre pour ramener les numéro de chambre et nous ramenons le nom service de l'objet service grâce au Deref(cham.cham_serv) qui nous permet d'y accéder.

Nous vérifions que l'élément pat.hospt est non null dans la table patient et nous vérifions que le bâtiment est le bâtiment B avec le Deref(cham.cham_serv) qui nous aide à accéder au bâtiment.

Puis nous vérifions que la mutuelle commence avec MN d'où le « LIKE » puis nous vérifions que le code service et le numéro chambre du patient est le même que celui de la table chambre, pour y accéder de la table patient on utilise le Deref(pat.hospt.hospt_cham) avec ça nous accédons à la table chambre que nous avons référencer dans le type thospitalisation.

```
SQL> SELECT pat.hospt.lit AS "numero lit",
2      cham.num_chambre AS "numero chambre",
3      Deref(cham.cham_serv).nom_service AS "nom service",
4      pat.nom_patient AS "nom patient",
5      pat.prenom_patient AS "prenom patient",
6      pat.mutuelle AS "mutuelle"
7 FROM patient pat, chambre cham
8 WHERE pat.mutuelle LIKE 'MN%' and Deref(cham.cham_serv).batiment = 'B'
9      and pat.hospt IS NOT NULL and
10      Deref(pat.hospt.hospt_cham).code_service = cham.code_service and
11      Deref(pat.hospt.hospt_cham).num_chambre = cham.num_chambre;
```

Figure 5.3 : Requete liste de patient qui commence par MN

-Le résultat

numero lit	numero chambre	nom service	

nom patient			

prenom patient			mutuelle

1	101	Cardiologie	
SERIR			
Mustapha			MNAM
3	101	Cardiologie	
TAHMI			
Lamia			MNH

numero lit numero chambre nom service			

nom patient			

prenom patient			mutuelle

2	105	Cardiologie	
TITOUCHE			
Mohamed			MNAM

Figure 5.4 : Resultat de la requete liste de patient qui commence par MN

Analyse des résultats

Nous remarquons qu'il y'a pas de patient dans la mutuelle commence par MN qui sont dans le service de cardiologie.

11 - Pour chaque patient soigné par plus de 3 médecins donner le nombre total de ses médecins ainsi que le nombre correspondant de spécialités médicales concernées.

```
SELECT pat.num_patient AS "NUM patient",  
pat.nom_patient AS "NOM patient",  
pat.prenom_patient AS "PRENOM patient",  
pat.nb_medecin() AS "nb_medecin",  
COUNT(DISTINCT Deref(patMed.column_value).specialite) AS "nb_specialite"  
FROM patient pat, TABLE(pat.pat_med) patMed  
GROUP BY pat.num_patient,  
pat.prenom_patient, pat.nom_patient, pat.nb_medecin()  
HAVING pat.nb_medecin() > 3;
```

Explication

Nous utilisons dans cette requête la table patient et la table imbriquée pat_med qui nous aide à accéder à la table médecin. Grâce à la table patient nous ramenons les attributs num patient, nom patient, le prénom du patient et le numéro des médecins qui l'ont soigné grâce à la méthode nb_medecin(), pour avoir le nombre de spécialités médicales prises on utilise le Deref sur la table pat_med avec la fonction value qui nous renvoie une instance de type médecin puis nous calculons le nombre de spécialités distinctes. Nous regroupons par num patient, nom patient, prénom patient et le nombre de médecin ou le nombre est supérieur à 3.

```
SQL> SELECT pat.num_patient AS "NUM patient",  
2 pat.nom_patient AS "NOM patient",  
3 pat.prenom_patient AS "PRENOM patient",  
4 pat.nb_medecin() AS "nb_medecin",  
5 COUNT(DISTINCT Deref(patMed.column_value).specialite) AS "nb_specialite"  
6 FROM patient pat, TABLE(pat.pat_med) patMed  
7 GROUP BY pat.num_patient,  
8 pat.prenom_patient, pat.nom_patient, pat.nb_medecin()  
9 HAVING pat.nb_medecin() > 3;
```

Figure 5.5 : Requête liste de patient soigné par 3 médecins

Le résultat

NUM patient	NOM patient		
PRENOM patient		nb_medecin	nb_specialite
6 ABERKANE	Aboukhalil	4	3
76 TECHTACHE	Noura	4	3
108 IDJAAD	Mohand	4	4
117 KECIR	Laziz	4	3
145 KEDJNANE	Brahim	4	4
147 BENNABI	Ahmed	4	4
159 MERABET	Ourida	4	4
172 ZERARGA	Mustapha	4	3

8 rows selected.

Figure 5.6 : Resultat de la requete liste de patient soigne par 3 medecins

Analyse des résultats

Nous remarquons qu'il n'y pas beaucoup de patients qui ont consulte plusieurs spécialités ou le max des spécialités consultes est de 4. On remarque aussi que les patients ont tendance à prendre l'avis d'un autre médecin de la même spécialité et ceci dans 50% des cas, En conclue qu'ils n'ont pas confiance en l'avis du médecin nous devrions prendre ça en considération.

12 - Quelle est la moyenne des salaires des infirmiers(ères) par service ?

```
SELECT serv.code_service AS "code_service",
serv.nom_service AS "nom service",
AVG(DEREF(servInf.column_value).salaire) AS "Moyenne par service"
FROM sservice serv, TABLE(serv.serv_inf) servInf
GROUP BY serv.code_service, serv.nom_service;
```

-Explication

Nous utilisons la table sservice et la table imbriquées serv_inf. De la table sservice nous ramenons le code service et le nom du service. On utilise le Deref sur la table serv_inf avec la fonction value qui nous renvoie une instance de type infirmier puis nous calculons la moyenne des salaires de chaque service avec la méthode AVG et nous groupons nos données par rapport au code service et nom service.

```
SQL> SELECT serv.code_service AS "code_service",
2 serv.nom_service AS "nom service",
3 AVG(DEREF(servInf.column_value).salaire) AS "Moyenne par service"
4 FROM sservice serv, TABLE(serv.serv_inf) servInf
5 GROUP BY serv.code_service, serv.nom_service;
```

Figure 5.7 : Requete Moyenne des salaires d'infirmiers.

-Le résultat de la requête

cod nom service	Moyenne par service
CHG Chirurgie générale	13076.9231
REA Réanimation et Traumatologie	11428.5714
CAR Cardiologie	12500

Figure 5.8 : Resultat de la requete moyenne des salaires d'infirmiers

Analyse des résultats

- Nous remarquons que le service ou les infirmiers sont le mieux payer est celui de la chirurgie générale suivie par ceux de la cardiologie. Pour la réanimation c'est moins bien rémunéré.

13 - Pour chaque service quel est le rapport entre le nombre d'infirmier(ères) affecté(es) au service et le nombre de patients hospitalisés dans le service ?

```
SELECT serv.code_service AS "code service",
serv.nom_service AS "nom service",
serv.nb_inf() AS "nb infirmier",
serv.nb_pat() AS "nb patient",
(serv.nb_inf() / serv.nb_pat()) AS "inf/pat"
FROM sservice serv;
```

Explication

Dans cette requête nous utilisons la table sservice seulement. Nous prenons le code service et le nom service de la table sservice. Pour le numéro de patient et le numéro d'infirmier nous utilisons les méthodes faites précédemment le nb_inf() et le nb_pat(), pour le rapport nous faisons la division du nb_inf() trouve précédemment sur le nb_pat().

```
SQL> SELECT serv.code_service AS "code service",
2 serv.nom_service AS "nom service",
3 serv.nb_inf() AS "nb infirmier",
4 serv.nb_pat() AS "nb patient",
5 (serv.nb_inf() / serv.nb_pat()) AS "inf/pat"
6 FROM sservice serv;
```

Figure 5.9 : Requete rapport infirmier patient

Le résultat de la requête

cod nom service	nb infirmier	nb patient
inf/pat		
CAR Cardiologie .88888889	8	9
REA Réanimation et Traumatologie .7	7	10
CHG Chirurgie générale .65	13	20

Figure 5.10 : Resultat de la requete rapport infirmier patient

Analyse des résultats

Nous remarquons des résultats que le service de chirurgie que le de patient est bien plus élevés que celui des infirmiers ceci pourrait engendrer un grand déficit si le nombre de patient augmente. Pour les services réanimation et cardiologie le nombre de patient n'est pas tres élevés par rapport au nombre d'infirmiers. L'hôpital devrait augmenter le nombre d'infirmiers pour le service de cardiologie.

14 - Donner la liste des médecins (Prénom et nom) ayant un patient hospitalisé dans chaque service.

```
SELECT med.num_emp AS "numero medecin",
med.nom_emp AS "nom medecin",
med.prenom_emp AS "prenom medecin"
FROM medecin med, TABLE(med.med_pat) medPat
WHERE Deref(medPat.column_value).hospt IS NOT NULL
GROUP BY med.num_emp,
med.nom_emp, med.prenom_emp
HAVING COUNT(
DISTINCT Deref(
Deref(medPat.column_value).hospt.hospt_cham).code_service) = (
SELECT COUNT(*) FROM sservice
);
```

Explication

Nous utilisons la table médecin et la table imbriquées med_pat. Avec la table médecin nous ramenons le numéro médecin, nom médecin, prénom médecin. Nous devons avoir le patient qui est hospitalisé pour ça nous vérifions avec le Deref sur la table pat_med avec la fonction value qui nous renvoie une instance de type patient puis nous faisons le hospt. Nous regroupons nos données sur le numéro médecin, nom médecin, prénom médecin ou le nombre de service est égale au nombre patient hospitalisé dans chacun d'eux. Pour faire cette dernière requête nous utilisons le Deref sur la table pat_med avec la fonction value qui nous renvoie une instance de type patient puis une nouvelle fois le Deref sur la table résultante du Deref précédente qui nous renvoie le code service de la chambre après avoir accéder à cette table puis nous comptons les valeurs distinctes.

```
SQL> SELECT med.num_emp AS "numero medecin",
2 med.nom_emp AS "nom medecin",
3 med.prenom_emp AS "prenom medecin"
4 FROM medecin med, TABLE(med.med_pat) medPat
5 WHERE Deref(medPat.column_value).hospt IS NOT NULL
6 GROUP BY med.num_emp,
7 med.nom_emp, med.prenom_emp
8 HAVING COUNT(
9 DISTINCT Deref(
10 Deref(medPat.column_value).hospt.hospt_cham).code_service) = (
11 SELECT COUNT(*) FROM sservice
12 );
```

Figure 5.11 : Requete liste de medecin

Le résultat de la requête

numero medecin	nom medecin	numero medecin	nom medecin
-----	-----	-----	-----
pre nom medecin		pre nom medecin	
-----		-----	
Ahcene	135 RAHALI	Mustapha	179 MOHAMMEDI
Fatima	99 BASSI	Naima	89 BAHBOUH
Hacine	144 BENDALI	Mohammed	126 BELGHALI

```
numero medecin nom medecin
-----
pre nom medecin
-----
Ahmed          31 ABDELAZIZ
Souad          85 BAALI
Nabila        196 TEBIBEL
9 rows selected.
```

Figure 5.12 : Resultat de la requete liste de medecin

Analyse des résultats

Des résultats obtenus nous remarquons qu'il y'a peu de médecins qui ont plusieurs patients hospitalisés dans plusieurs services donc il n'y a pas beaucoup de charge sur les médecins ce qui est bon pour eux et que les services sont bien répertoriés par l'hôpital.