EXO I /12 pt.

1. A quoi sert la syntaxe : *CREATE OR REPLACE TYPE*

Créer un nouveau type s'il n'existe pas, c'est si le contraire il sera créé sans avoir recours à le supprimer.

2. Pour quels cas utilisons-nous la fonction IS OF ? donner un exemple.

La fonction SQL IS OF (nomType[,monType]) renvoie TRUE si la référence testée appartient au (x) type(s) listé(s) en paramètre.

Exemple

```
CREATE TYPE Pilote_type AS OBJECT
(brevet VARCHAR(6), nom VARCHAR(20))
NOT FINAL
/
CREATE TYPE PiloteChasse_type
 UNDER Pilote_type (baseAerienne VARCHAR(20),
 surnom VARCHAR(10))
/
CREATE TYPE Avion_type AS OBJECT
        (immat VARCHAR(6), ref_pilote REF Pilote_type)
/
CREATE TABLE Avion OF Avion_type
        (CONSTRAINT pk_Avion PRIMARY KEY(immat));

CREATE TABLE Pilote OF Pilote_type
        (CONSTRAINT pk_Pilote PRIMARY KEY(brevet));

CREATE TABLE PiloteChasse OF PiloteChasse_type
        (CONSTRAINT pk_PiloteChasse PRIMARY KEY(brevet));

INSERT INTO Pilote VALUES (Pilote_type('PL-1','Miranda'));
INSERT INTO PiloteChasse VALUES (PiloteChasse_type('PL-3','Soutou','Mr de Marsan','Faucon'));

INSERT INTO Avion VALUES
        (Avion_type('F-WTSS',(SELECT REF(p) FROM Pilote p WHERE p.brevet='PL-1')));

INSERT INTO Avion VALUES
        (Avion_type('F-PAUL', (SELECT REF(p) FROM PiloteChasse p WHERE p.brevet='PL-3')));

SELECT a.immat,a.ref_pilote.brevet, a.ref_pilote.nom  FROM Avion a WHERE DEREF(a.ref_pilote) IS
OF (PiloteChasse_type);
```

3. Pour quels cas utilisons-nous la fonction TREAT ? donner un exemple.

Il est possible de sélectionner des références à un niveau donné d'une hiérarchie. La fonction TREAT permet de programmer cette dichotomie en accédant à une référence substituable.

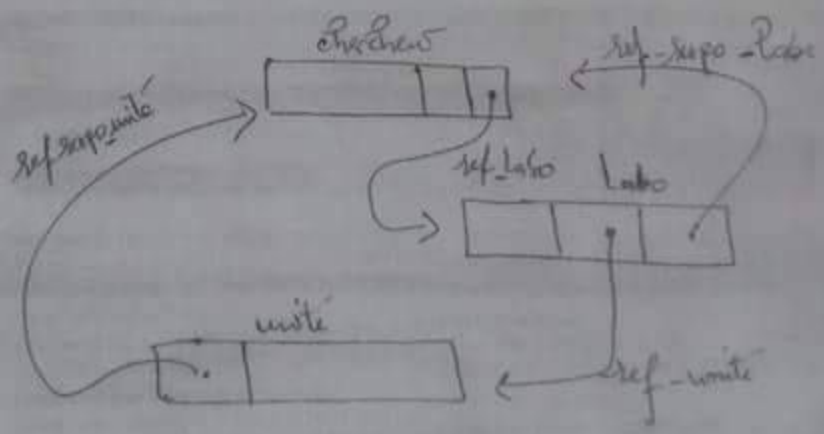
```
SELECT TREAT(ref_pilote AS REF PiloteChasse_type) "REF Chasseurs" FROM Avion;
```

9. Créer les types correspondants à cette description

Un chercheur appartient à un laboratoire de recherche qui est à son tour affilié à une unité de recherche. Chaque labo possède un responsable qui est un chercheur, même chose pour une unité.

Attributs : idcherch, nomcherch, precherch, nomlab, idlab, nomunit, idunit.

```
CREATE TYPE chercheur_type ;  CREATE TYPE unitederecherche_type ;

CREATE TYPE labo_type AS OBJECT (
idlab NUMBER, nomlab VARCHAR2(50),
refrespolab REF chercheur,
refunitederecherche REF unitederecherche_type)
/

CREATE OR REPLACE TYPE  chercheur_type  AS OBJECT
(idcherch       NUMBER,
nomcherch       VARCHAR2(30),
precherch       VARCHAR2(30),
ref_labo        labo_type)
/

CREATE OR REPLACE TYPE unitederecherche_type  AS OBJECT (
idunit NUMBER,
nomunit VARCHAR2(50),
refresponunite REF chercheur_type,
)
/
CREATE TABLE chercheurs  OF chercheur_type
(CONSTRAINT pk_chercheur PRIMARY KEY(idcherch));
CREATE TABLE laboratoires OF labo_type
        (CONSTRAINT pk_labo PRIMARY KEY(idlabo));
CREATE TABLE unitederecherche OF unitederecherche_type
        (CONSTRAINT pk_unite PRIMARY KEY(idunit));   (RQ : vous pouvez aussi déclarer des contraintes.)
```



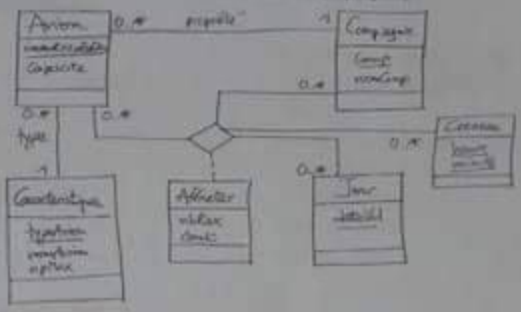
7. Donner un exemple avec la déclaration d'une fonction membre et d'une procédure membre

```
CREATE TYPE Compagnie_type AS OBJECT
(comp VARCHAR(4), siege VARCHAR(20),
 MEMBER FUNCTION  compteFlotte  RETURN NUMBER,
MEMBER PROCEDURE demenage   (a IN Siege_social_type))
/

CREATE TYPE Avion_type AS OBJECT
(immat VARCHAR2(6),  ref_Compagnie REF Compagnie_type)
/

CREATE OR REPLACE TYPE BODY Compagnie_type AS
MEMBER FUNCTION compteFlotte RETURN NUMBER IS
resultat NUMBER;
 BEGIN
        SELECT COUNT (immat) INTO resultat FROM Avion a
                WHERE a.ref_Compagnie.comp = SELF.comp;
        RETURN resultat;
 END compteFlotte;
--
MEMBER PROCEDURE demenage (a IN Siege_social_type) IS
 BEGIN
        UPDATE Compagnie
                SET siege_social = a
                WHERE comp = SELF.comp;
 END demenage;
END;
/
```

8. Donner un exemple (SQL 3) illustrant l'héritage multiple.

L'héritage multiple n'est pas autorisé en sql3 oracle.

10. Proposer un schéma RO du modèle de classes suivant (expliquer)



```
CREATE TYPE Compagnie_type AS OBJECT
 (comp VARCHAR(4), nomComp VARCHAR(15))
/

CREATE TYPE Avion_type AS OBJECT
(immatriculation VARCHAR(6), capacite NUMBER(3), type REF Car_type,
comp REF compagnie_type)
/

CREATE TYPE car_type AS OBJECT
 (typeAvion VARCHAR(6), nomavion VARCHAR(15), npmax NUMBER(3))
/

CREATE TYPE afreter_type AS OBJECT
(Ref_avion      REF avion_type,
Ref_compagnie REF compagnie_type,
Datevol DATE, heure NUMBER(2), minute NUMBER (2),
nbpax NUMBER(2), cout NUMBER(6,2))
/

CREATE TABLE car  OF car_type
        (CONSTRAINT pk_typeav PRIMARY KEY(typeAvion));

CREATE TABLE avion OF avion_type
(ref_Compagnie REF Compagnie_type CONSTRAINT consref_Comp REFERENCES Compagnie,
ref_Compagnie REF Car_type CONSTRAINT consref_Car REFERENCES Car,
CONSTRAINT pk_Avion PRIMARY KEY(immatriculation),
CONSTRAINT nn_ref_Compagnie CHECK (ref_Compagnie IS NOT NULL),
CONSTRAINT nn_ref_Car CHECK (ref_Car IS NOT NULL);

CREATE TABLE compagnie  OF compagnie_type
        (CONSTRAINT pk_compagnie PRIMARY KEY(comp));

CREATE TABLE affreter  OF affreter_type (
CONSTRAINT nn_refAvi CHECK (re_fAvion IS NOT NULL),
CONSTRAINT fk_refAvi    ref_Avion REFERENCES Avion,

CONSTRAINT nn_ref_comp CHECK(ref_compagnie IS NOT NULL),
CONSTRAINT fk_ref_comp REFERENCES compagnie,

CONSTRAINT nn_jour (datevol IS NOT NULL),
CONSTRAINT nn_heure CHECK (heure IS NOT NULL),
CONSTRAINT nn_minute CHECK (minute IS NOT NULL));
```