**AZEAU Yannick**
**MONTET David**

28/09/2004

Ntie31

# Compte Rendu de TP de Base de donnée SQL3

A l'Attention de M. SOUTOU

*Maîtrise IUP NITE*

# Sommaire

# TP n°1 Types et Tables SQL3

## 1. Création de la Base de données

Dans cette étude, on se propose de construire une base de données en SQL3 d'un système de gestion d'avions. Chaque avion appartient à une compagnie.

### 1.1 Script de suppression des tables et des types

Il faut détruire les tables dans l'ordre inverse de création afin de ne pas violer les contraintes de `REF`. Pour les types, un ordre doit aussi être respecté.

```
DROP TABLE avion_sql3;
DROP TABLE typeAvion_sql3;
DROP TABLE compagnie_sql3;

DROP TYPE avion_type;
DROP TYPE typeAvion_type;
DROP TYPE affreter_nested;
DROP TYPE elt_affreter_nested;
DROP TYPE compagnie_type;
```

### 1.2 Script de création des types

Il faut d'abord créer les types de données avant de créer les tables car ces dernières sont créées à partir des types.

Le langage SQL3 permet de créer ses propres types de données par l'intermédiaire de l'instruction de CREATE TYPE.

Cet objet ainsi créé devient utilisable dans une base de données par l'intermédaire d'une définition de type habituelle. Dans ce cas, le champ typé pourrait accepter plusieurs zones de types différents.

```
CREATE TYPE compagnie_type AS OBJECT
       (comp VARCHAR(4),
        nom_comp VARCHAR(30),
        nb_av NUMBER(3))
/

CREATE TYPE elt_affreter_nested AS OBJECT
      (ref_comp_sql3 REF compagnie_type,
       date_a DATE,
       nb_p NUMBER(3),
       cout NUMBER (8))
/

-- Creation de la collection : liste d'affretements
CREATE TYPE affreter_nested AS TABLE OF elt_affreter_nested
/

CREATE OR REPLACE TYPE typeAvion_type AS OBJECT
      ( typa VARCHAR(4),
        npmax NUMBER(3),
        nomAvion VARCHAR(30),
        nb_av NUMBER(3))
/

CREATE TYPE avion_type AS OBJECT
      ( na VARCHAR(4),
        ref_typa_sql3 REF typeAvion_type,
```

```
        cap NUMBER(3),
        ref_comp_sql3  REF compagnie_type,
        collec affreter_nested )
/
```

### 1.3 Script de création des tables avec les contraintes

```
CREATE TABLE compagnie_sql3 OF compagnie_type
( CONSTRAINT pk_compagnie PRIMARY KEY (comp));

CREATE TABLE typeAvion_sql3 OF typeAvion_type
(   CONSTRAINT pk_typeAvion_sql3 PRIMARY KEY ( typa));

CREATE TABLE avion_sql3 OF avion_type
(
     CONSTRAINT pk_avion_sql3 PRIMARY KEY (na),
     CONSTRAINT contRef ref_comp_sql3  REFERENCES compagnie_sql3,
     CONSTRAINT contRef ref_typa_sql3  REFERENCES typeAvion_sql3,
     CONSTRAINT ck_cap CHECK (cap between 0 AND 500),
     CONSTRAINT ck_comp CHECK (ref_comp_sql3 IS NOT NULL))
     NESTED TABLE collec STORE AS tabAffreter;
```

### 1.4 Vérification de la bonne création des types et des tables

```
DESC compagnie_type

 Nom                                       NULL ?   Type
 ----------------------------------------- -------- ----------------
 COMP                                               VARCHAR2(4)
 NOM_COMP                                           VARCHAR2(30)
 NB_AV                                              NUMBER(3)

DESC elt_affreter_nested

 Nom                                       NULL ?   Type
 ----------------------------------------- -------- ---------------------
 REF_COMP_SQL3                                      REF OF COMPAGNIE_TYPE
 DATE_A                                             DATE
 NB_P                                               NUMBER(3)
       COUT                                         NUMBER(8)

DESC affreter_nested


 affreter_nested TABLE OF ELT_AFFRETER_NESTED
 Nom                                       NULL ?   Type
 ----------------------------------------- -------- ---------------------------
 REF_COMP_SQL3                                      REF OF COMPAGNIE_TYPE
 DATE_A                                             DATE
 NB_P                                               NUMBER(3)
 COUT                                               NUMBER(8)

DESC typeAvion_type

 Nom                                       NULL ?   Type
 ----------------------------------------- -------- -----------------
 TYPA                                               VARCHAR2(4)
 NPMAX                                              NUMBER(3)
 NOMAVION                                           VARCHAR2(30)
 NB_AV                                              NUMBER(3)


DESC avion_type

 Nom                                       NULL ?   Type
 ----------------------------------------- -------- ---------------------------
 NA                                                 VARCHAR2(4)
```

```
 REF_TYPA_SQL3                                      REF OF TYPEAVION_TYPE
 CAP                                                NUMBER(3)
 REF_COMP_SQL3                                      REF OF COMPAGNIE_TYPE
 COLLEC                                             AFFRETER_NESTED
```

```
DESC compagnie_sql3

 Nom                                       NULL ?   Type
 ----------------------------------------- -------- ------------------
 COMP                                      NOT NULL VARCHAR2(4)
 NOM_COMP                                           VARCHAR2(30)
 NB_AV                                              NUMBER(3)

DESC typeAvion_sql3
 Nom                                       NULL ?   Type
 ----------------------------------------- -------- ------------------
 TYPA                                      NOT NULL VARCHAR2(4)
 NPMAX                                              NUMBER(3)
 NOMAVION                                           VARCHAR2(30)
 NB_AV                                              NUMBER(3)


DESC avion_sql3

 Nom                                     NULL ?   Type
 --------------------------------------- -------- ----------------------------
 NA                                      NOT NULL VARCHAR2(4)
 REF_TYPA_SQL3                                    REF OF TYPEAVION_TYPE
 CAP                                              NUMBER(3)
 REF_COMP_SQL3                                    REF OF COMPAGNIE_TYPE
 COLLEC                                           AFFRETER_NESTED
```

Les types que l'on a créés sont par défaut en not final et instanciables

On vérifie dans les vues que les types ont été créés. On voit qu'ils ont bien été créé en tant qu'objet  et par défaut YES sur Final et instanciable

```
SELECT type_name, type_oid, typecode,  final, instantiable FROM USER_TYPES;

TYPE_NAME                      TYPE_OID
------------------------------ --------------------------------
TYPECODE                       FIN INS
------------------------------ --- ---
ADR_TYPE                       E526D740F77F4BD6E03400306E06F2B9
OBJECT                         NO  YES

AFFRETER_NESTED                E445550E35AD5BC3E03400306E06F2B9
COLLECTION                     YES YES

AVION_TYPE                     E445550E35B85BC3E03400306E06F2B9
OBJECT                         YES YES

COMPAGNIE_TYPE                 E445550E35A05BC3E03400306E06F2B9
OBJECT                         YES YES

COMP_VIRTUELLE_TYPE            E526D740FA034BD6E03400306E06F2B9
OBJECT                         NO  NO

ELT_AFFRETER_NESTED            E445550E35A65BC3E03400306E06F2B9
OBJECT                         YES YES

ELT_EMPLOYEURS_NT_TYPE         E49639AD084E3999E03400306E06F2B9
OBJECT                         YES YES

ELT_QUALIFS_VRY_TYPE           E49639AD08D63999E03400306E06F2B9
OBJECT                         YES YES

EMPLOYEURS_NT_TYPE             E49639AD085A3999E03400306E06F2B9
```

```
COLLECTION                     YES YES

PILOTE_TYPE                    E49639AD0AA93999E03400306E06F2B9
OBJECT                         YES YES

TYPE_NAME                      TYPE_OID
------------------------------ --------------------------------
TYPECODE                       FIN INS
------------------------------ --- ---
QUALIFS_VRY_TYPE               E49639AD08283999E03400306E06F2B9
COLLECTION                     YES YES

TYPEAVION_TYPE                 E445550E35B15BC3E03400306E06F2B9
OBJECT                         YES YES
```

```
SELECT table_name, table_type FROM USER_OBJECT_TABLES;
```

```
TABLE_NAME                     TABLE_TYPE
------------------------------ ------------------------------
AVION_SQL3                     AVION_TYPE
COMPAGNIE_SQL3                 COMPAGNIE_TYPE
EMPLOYEURS                     ELT_EMPLOYEURS_NT_TYPE
PILOTESQL3                     PILOTE_TYPE
TABAFFRETER                    ELT_AFFRETER_NESTED
TYPEAVION_SQL3                 TYPEAVION_TYPE
```

### *1.5 Insertion d'enregistrements*

Insertion dans la table compagnie_sql3

```
INSERT INTO compagnie_SQL3 VALUES ('AF','Air France',4);
INSERT INTO compagnie_SQL3 VALUES ('TAT','Transport Air Touraine',2);
INSERT INTO compagnie_SQL3 VALUES ('AL','Air Liberté',0);
INSERT INTO compagnie_SQL3 VALUES ('AOM','Air Outre-Mer',0);
```

Insertion dans la table typeavion_sql3

```
INSERT INTO typeavion_SQL3 VALUES ('B747',400,'Boeing Jumbo 747',1);
INSERT INTO typeavion_SQL3 VALUES ('A320',180,'Airbus A320',2);
INSERT INTO typeavion_SQL3 VALUES ('A310',250,'Airbus A310',1);
INSERT INTO typeavion_SQL3 VALUES ('B727',150,'Boeing 727',2);
INSERT INTO typeavion_SQL3 VALUES ('CRJ1',180,'Canadian Regional Jet
1',0);
```

Insertion dans la table avion_sql3

On utilise l'opérateur REF pour insérer des références. Celui-ci renvoi la référence (un OID) d'un enregistrement.

```
INSERT INTO avion_SQL3
      VALUES (avion_type
            ('A1',(SELECT REF(t) FROM typeavion_SQL3 t WHERE t.typa='A320'),
              170,(SELECT REF(c) FROM compagnie_SQL3 c WHERE c.comp='AF'),
              affreter_nested
                  (affreter_elt_nested((SELECT REF(c) FROM compagnie_SQL3 c
WHERE c.comp='TAT'),
                              '10-12-2000',120,75000),
                   affreter_elt_nested((SELECT REF(c) FROM compagnie_SQL3 c
WHERE c.comp='AL'),
                              '11-12-2000',150,87000),
                   affreter_elt_nested((SELECT REF(c) FROM compagnie_SQL3 c
WHERE c.comp='AF'),
```

```
                              '09-12-2000',110,71000),
                   affreter_elt_nested((SELECT REF(c) FROM compagnie_SQL3 c
WHERE c.comp='AF'),
                              '19-02-2001',120,73000) ) ));

INSERT INTO avion_SQL3
      VALUES (avion_type
            ('A2',(SELECT REF(t) FROM typeavion_SQL3 t WHERE t.typa='B727'),
              140,(SELECT REF(c) FROM compagnie_SQL3 c WHERE c.comp='TAT'),
              affreter_nested() ));

INSERT INTO avion_SQL3
      VALUES (avion_type
            ('A3',(SELECT REF(t) FROM typeavion_SQL3 t WHERE t.typa='B747'),
              400,(SELECT REF(c) FROM compagnie_SQL3 c WHERE c.comp='AF'),
              affreter_nested
                  (affreter_elt_nested((SELECT REF(c) FROM compagnie_SQL3 c
WHERE c.comp='AF'),
                              '09-12-2000',300,185000) )));

INSERT INTO avion_SQL3
      VALUES (avion_type
            ('A4',(SELECT REF(t) FROM typeavion_SQL3 t WHERE t.typa='B727'),
              120,(SELECT REF(c) FROM compagnie_SQL3 c WHERE c.comp='AF'),
              affreter_nested() ));

INSERT INTO avion_SQL3
      VALUES (avion_type
            ('A5',(SELECT REF(t) FROM typeavion_SQL3 t WHERE t.typa='A310'),
              200,(SELECT REF(c) FROM compagnie_SQL3 c WHERE c.comp='TAT'),
              affreter_nested() ));

INSERT INTO avion_SQL3
      VALUES (avion_type
            ('A6',(SELECT REF(t) FROM typeavion_SQL3 t WHERE t.typa='A320'),
              160,(SELECT REF(c) FROM compagnie_SQL3 c WHERE c.comp='AF'),
              affreter_nested() ));
```

# TP n°2

## 2. Interrogation et manipulation de collections

### 2.1 Préliminaires :

Création des types nécessaires à la table Pilote.

Comme pour la création des types des tables précédentes, si on désire lancer le script plusieurs fois, il faut supprimer la table en premier lieu puis les types. Les qualifications seront stockés dans un VARRAY et les employeurs dans une NESTED TABLE.

```
DROP TABLE PiloteSQL3;

DROP TYPE elt_qualifs_vry_type;
DROP TYPE elt_employeurs_nt_type;
DROP TYPE qualifs_vry_type ;
DROP TYPE employeurs_nt_type;

CREATE OR REPLACE TYPE elt_employeurs_nt_type AS OBJECT
(  comp VARCHAR(4),
   duree NUMBER(3)
)
/

CREATE TYPE elt_employeurs_nt_type  AS TABLE OF elt_employeurs_nt_type
/

CREATE TYPE  elt_qualifs_vry_type AS OBJECT
(   typeav   VARCHAR(4),
    datequal DATE
)
/

CREATE TYPE qualifs_vry_type AS VARRAY(3) OF elt_qualifs_vry_type
/

CREATE OR REPLACE TYPE pilote_type AS OBJECT
(
   brevet          VARCHAR(4),
   nom             VARCHAR(20),
   qualifs_vry     qualifs_vry_type,
   employeurs_nt   employeurs_nt_type
)
/
```

Création de la table Pilote.

```
CREATE TABLE PiloteSQL3 OF pilote_type
( CONSTRAINT pk_piloteSQL3 PRIMARY KEY (brevet))
NESTED TABLE employeurs_nt STORE AS employeurs;

DESC piloteSQL3 p;

 Nom                                       NULL ?   Type
 ----------------------------------------- -------- ----------------------------
 BREVET                                    NOT NULL VARCHAR2(4)
 NOM                                                VARCHAR2(20)
 QUALIFS_VRY                                        QUALIFS_VRY_TYPE
 EMPLOYEURS_NT                                       EMPLOYEURS_NT_TYPE
```

## 2.2 *Mise à jour dans une nested table*

### 2.2.1 Insertion d'un pilote p1 n'ayant pas de qualification mais 3 employeurs.

Pour créer un enregistrement de qualification vide, on utilise le constructeur du type des qualifications. On pourra ainsi rajouter plus tard des qualifications.
Pour ajouter 3 éléments dans la liste des employeurs, on utilise 3 fois le constructeur de l'élément de la liste : `elt_employeurs_nt_type`

```
INSERT INTO piloteSQL3 VALUES ('p1','Tanguy',qualifs_vry_type(),
      employeurs_nt_type ( elt_employeurs_nt_type('AF',30),
                           elt_employeurs_nt_type('TAT',20),
                           elt_employeurs_nt_type('TA',150) )       );
```

### 2.2.2 Interrogation de la collection employeurs du pilote p1

L'opérateur TABLE permet de considérer que la collection peut être traitée comme une table.

```
SELECT a.comp, a.duree  FROM TABLE(SELECT employeurs_nt FROM piloteSQL3 p
                                   WHERE p.brevet='p1') a;

COMP      DUREE
---- ----------
AF           30
TAT          20
TA          150
```

### 2.2.3 Insertion d'un pilote p2 n'ayant jamais de qualification et pas encore d'employeurs.

On donne la valeur NULL à la collection qui ne prendra jamais de valeur. L'ajout et la modification de qualification est impossible. Par contre, la collection des employeurs est vide et pourra être modifiée.

```
INSERT INTO piloteSQL3
      VALUES ('p2','Laverdure',NULL,employeurs_nt_type ());
```

### 2.2.4 Interrogation de la collection employeurs du pilote p2

```
SELECT a.comp, a.duree
   FROM TABLE(SELECT employeurs_nt FROM piloteSQL3 p WHERE p.brevet='p2')
a;

aucune ligne sélectionnée
```

Cet ordre ne renvoie aucune ligne puisque la collection des employeurs est vide.

### 2.2.5 Insertion de 2 employeurs à p2

```
INSERT INTO
      TABLE( SELECT employeurs_nt FROM piloteSQL3 WHERE brevet='p2' )
      VALUES (elt_employeurs_nt_type ('AF',30));

INSERT INTO
      TABLE( SELECT employeurs_nt FROM piloteSQL3 WHERE brevet='p2' )
      VALUES (elt_employeurs_nt_type ('TAT',15));
```

Remarque : ne pas oublier les parenthèses dans le VALUE.

### 2.2.6 Interrogation de la collection employeurs du pilote p2

```
SELECT a.comp, a.duree
FROM TABLE(SELECT employeurs_nt FROM piloteSQL3 p WHERE p.brevet='p2') a;

COMP  DUREE
----  -----
AF    30
TAT   15
```

### 2.2.7 Modification de la collection employeurs du pilote p2

On augmente de 20% la duree du contrat de la compagnie 'AF' de p2.
Pour cela, on utilise une fois de plus l'opérateur TABLE.

```
UPDATE TABLE(SELECT employeurs_nt FROM piloteSQL3 p WHERE p.brevet='p2') a
      SET a.duree = a.duree + 0.2*a.duree
      WHERE a.comp = 'AF';

1 ligne mise à jour.
```

### 2.2.8 Suppression de l'employeur 'AF' du pilote p2

```
DELETE FROM
      TABLE(SELECT employeurs_nt FROM piloteSQL3 WHERE brevet='p2') a
      WHERE a.comp='AF';

1 ligne supprimée.
```

Vérification :

```
SELECT a.comp, a.duree
FROM TABLE(SELECT employeurs_nt FROM piloteSQL3 p WHERE p.brevet='p2') a;

COMP  DUREE
----  -----
TAT   15
```

## *2.3 Mise à jour dans un varray*

### 2.3.1 Insertion du pilote p3

Le pilote n'a pour l'instant aucune qualifications, ni employeurs. On utilise donc le constructeur pour initialiser à vide chaque collection avec la possibilité de rajouter des élements plus tard.

```
INSERT INTO piloteSQL3
VALUES ('p3','Daurat', qualifs_vry_type(), employeurs_nt_type());
```

### 2.3.2 Insertion de 2 qualifications pour p3

Ici, l'opérateur TABLE ne fonctionne pas pour un VARRAY. L'erreur est normale.

```
INSERT INTO TABLE(SELECT qualifs_vry  FROM piloteSQL3 WHERE brevet='p3')
VALUES (elt_qualifs_vry_type ( 'A320','31-12-2004'));

ERREUR à la ligne 1 :
```

```
ORA-25015: cannot perform DML on this nested TABLE view column
```

On retiendra donc la solution suivante au moyen d'un bloc PL/SQL :

```
DECLARE
  temp qualifs_vry_type;
  ta REF typeavion_type;

BEGIN
  -- Chargement du tableau dans une variable temporaire
  SELECT qualifs_vry INTO temp
      FROM piloteSQL3
      WHERE brevet='p3' FOR UPDATE;

  SELECT REF(t) INTO ta  FROM typeavion_sql3 t WHERE t.typa='A320';

  --Ajout d'une qualification dans temp
  temp.EXTEND;
  temp(1) := elt_qualifs_vry_type(ta,'31-12-2004');

  SELECT REF(t) INTO ta  FROM typeavion_sql3 t WHERE t.typa='A330';

  --Ajout d'une qualification dans temp
  temp.EXTEND;
  temp(2) := elt_qualifs_vry_type(ta,'15-05-2003');

  -- Mide à jour de la collection qualifs_vry
  UPDATE piloteSQL3
      SET qualifs_vry = temp
      WHERE brevet='p3';
END;
/

Procédure PL/SQL terminée avec succès.
```

Pour insérer des éléments dans un VARRAY, on note donc 3 étapes :

1. Charger le tableau dans une variable temporaire
2. Rajouter à cette variable des éléments ( méthode EXTEND)
3. Mettre à jour la collection à partir de la variable temporaire

### 2.3.3 Interrogation de la qualification de p3

```
SELECT qvt.typeav, qvt.datequal
 FROM TABLE(SELECT qualifs_vry FROM piloteSQL3 p WHERE p.brevet='p3') qvt;

TYPE DATEQUAL
---- --------
A320 31/12/04
A330 15/05/03
```

La collection a bien été mise à jour avec les données ci-dessus.

### 2.3.4 Modification d'un varray

```
UPDATE TABLE ( SELECT qualifs_vry FROM piloteSQL3 WHERE brevet='p3') p
SET p.datequal='13-05-2003'
WHERE p.typeav='A330';

*
ERREUR à la ligne 2 :
ORA-25015: cannot perform DML on this nested TABLE view column
```

Oracle génère une erreur. On ne peut donc pas aujourd'huimodifier un VARRAY avec l'instruction UPDATE couplée avec l'opérateur TABLE. Il faut écrire un bloc PL/SQL. On distingue 2 cas :

1er cas : on connait l'indice du tableau

```
DECLARE
  temp qualifs_vry_type;
  ta REF typeavion_type;

BEGIN

  SELECT qualifs_vry INTO temp
      FROM piloteSQL3
      WHERE brevet='p3' FOR UPDATE;

  SELECT REF(t) INTO ta  FROM typeavion_sql3 t WHERE t.typa='A330';

  temp(2).datequal:='13-05-2003';
  temp.EXTEND;

  UPDATE piloteSQL3
            SET qualifs_vry = temp
            WHERE brevet='p3';
END;
/

Procédure PL/SQL terminée avec succès.
```

2ème cas : on ne connait l'indice du tableau

```
DECLARE
  temp qualifs_vry_type;

BEGIN

  SELECT qualifs_vry INTO temp
      FROM piloteSQL3  p
      WHERE p.brevet='p3' FOR UPDATE;

  temp(temp.COUNT+1).datequal='13-05-2003';
  temp.EXTEND;

  UPDATE piloteSQL3
            SET qualifs_vry = temp
            WHERE brevet='p3';
END;
/

Procédure PL/SQL terminée avec succès.
```

## 2.3.5 Dépassement de capacité d'un varray

```
DECLARE
  temp qualifs_vry_type;
  ta REF typeavion_type;
BEGIN
  SELECT qualifs_vry INTO temp
      FROM piloteSQL3
      WHERE brevet='p3' FOR UPDATE;

  temp.EXTEND;
```

```
  temp(3) := elt_qualifs_vry_type('B727',SYSDATE);

  -- Depassement de la capacite, erreur
  temp.EXTEND;
  temp(4) := elt_qualifs_vry_type('B728',SYSDATE);

  UPDATE piloteSQL3  SET qualifs_vry = temp
      WHERE brevet='p3';
END ;
/

*
ERREUR à la ligne 1 :
ORA-06532: Subscript outside of limit
ORA-06512: at line 15
```

Oracle lève une erreur car la taille du varray était limitée à 3.

### 2.3.6 Ajout de la qualification B727

```
DECLARE
  temp qualifs_vry_type;
  ta REF typeavion_type;
BEGIN
  SELECT qualifs_vry INTO temp
      FROM piloteSQL3
      WHERE brevet='p3' FOR UPDATE;

  temp.EXTEND;
  temp(3) := elt_qualifs_vry_type('B727',SYSDATE);


UPDATE piloteSQL3  SET qualifs_vry = temp
      WHERE brevet='p3';
END;
/
Procédure PL/SQL terminée avec succès.
```

Vérification :

```
SELECT qvt.typeav, qvt.datequal
 FROM TABLE(SELECT qualifs_vry FROM piloteSQL3 p WHERE p.brevet='p3') qvt;

TYPE DATEQUAL
---- --------
A320 31/12/04
A330 15/05/03
B727 25/05/03
```

### 2.3.7 Suppression d'éléments dans un VARRAY

Le DELETE ne marche pas pour un varray. Il faut mettre à jour un tableau vide par un bloc PL / SQL. La fonction disponible TRIM permet de supprimer des éléments d'une collection à partir de la fin.

```
DECLARE
  temp qualifs_vry_type;
  ta REF typeavion_type;
BEGIN
  SELECT qualifs_vry INTO temp
            FROM piloteSQL3
            WHERE brevet='p3' FOR UPDATE;
```

```
  -- Supprime 2 qualifications du tableau
  temp.TRIM(2);

  UPDATE piloteSQL3  SET qualifs_vry = temp
      WHERE brevet='p3';
END ;
/

Procédure PL/SQL terminée avec succès.
```

Vérification :

```
SELECT qvt.typeav, qvt.datequal
 FROM TABLE(SELECT qualifs_vry FROM piloteSQL3 p WHERE p.brevet='p3') qvt;

aucune ligne sélectionnée.
```

## *2.4 Utilisation des pointeurs*

### 2.4.1 Compagnie(code, nom) propriétaire d'au moins 1 avion

```
SELECT DISTINCT a.ref_comp_sql3.comp , a.ref_comp_sql3.nom_comp
      FROM Avion_sql3 a;

REF_ REF_COMP_SQL3.NOM_COMP
---- ------------------------------
AF   Air France
TAT  Transport Air Touraine
```

### 2.4.2 Immatriculation des avions de la compagnie de code TAT

```
SELECT  a.na FROM avion_sql3 a WHERE a.ref_comp_sql3.comp='TAT';

*NA
----
A2
A5
```

### 2.4.3 Immatriculation et nom des avions construits à Blagnac

```
SELECT a.na, a.ref_typa_sql3.nomAvion
      FROM avion_sql3 a
      WHERE UPPER (a.ref_typa_sql3.nomavion) LIKE '%AIRBUS%';

NA   REF_TYPA_SQL3.NOMAVION
---- ------------------------------
A1   Airbus A320
A5   Airbus A310
A6   Airbus A320
```

### 2.4.4 Immatriculation et nom des avions appartenant à la compagnie 'AF'

```
SELECT a.na, a.ref_typa_sql3.nomAvion
      FROM avion_sql3 a
      WHERE a.ref_comp_sql3.comp = 'AF';

NA   REF_TYPA_SQL3.NOMAVION
---- ------------------------------
A1   Airbus A320
A3   Boeing Jumbo 747
A4   Boeing 727
A6   Airbus A320
```

## *2.5 Utilisation de fonctions*

### 2.5.1 Plus grande et plus petite capacité des avions

```
SELECT min(a.cap) MINIMUM, MAX(a.cap) MAXIMUM FROM avion_sql3 a;


   MINIMUM    MAXIMUM
---------- ----------
       120        400
```

### 2.5.2 Immatriculation, capacité et nom de l'avion ayant la plus grande capacité

```
SELECT a.na, a.cap, a.ref_typa_sql3.nomAvion
      FROM avion_sql3 a
      WHERE a.cap = (SELECT MAX(a.cap) FROM avion_sql3 a);

NA CAP    NOMAVION
-- ------ -----------------------
A5 200    Airbus A310
```

### 2.5.3 Capacité moyenne des avions d'Air France

```
SELECT ROUND(AVG(a.cap),0) Capacite_moyenne FROM avion_sql3 a;

CAPACITE_MOYENNE
----------------
             198
```

### 2.5.4 Capacité totale de la flotte de TAT

```
SELECT SUM(a.cap) Capacite_Totale_TAT
      FROM avion_sql3 a
      WHERE a.ref_comp_sql3.comp='TAT';

CAPACITE_TOTALE_TAT
-------------------
                340
```

### 2.5.5 Avions d'Air Francee dont la capacité dépasse la moyenne des avions d'Air France

```
SELECT a.na, a.ref_typa_sql3.nomAvion FROM avion_sql3 a
      WHERE cap > (SELECT AVG(av.cap) FROM avion_sql3 av
                          WHERE av.ref_comp_sql3.comp='AF');

NA   REF_TYPA_SQL3.NOMAVION
---- ------------------------------
A3   Boeing Jumbo 747
```

### 2.5.6 Nombre d'avions possédés par la compagnie Air France

```
SELECT COUNT(*) Nb_Avion_AF
      FROM avion_sql3 a
      WHERE a.ref_comp_sql3.comp='AF';

NB_AVION_AF
-----------
          4
```

### 2.5.7 Nombre d'avions pour chaque compagnie (sans utiliser la colonne nb_av). Afficher aussi ensuirte les compagnies ne possédant pas d'avion.

```
SELECT a.ref_comp_sql3.comp, count(*) nb_avion FROM avion_sql3 a
      GROUP BY a.ref_comp_sql3.comp
UNION
SELECT c.comp, 0 nb_avion FROM compagnie_sql3 c  WHERE comp NOT IN
     (SELECT a.ref_comp_sql3.comp FROM avion_sql3 a );


REF_   NB_AVION
---- ----------
AF            4
AL            0
AOM           0
TAT           2
```

### 2.5.8 Compagnie(s) ayant plus d'un avion de même type

```
SELECT a.ref_comp_sql3.comp FROM avion_sql3 a
      GROUP BY a.ref_comp_sql3.comp,a.ref_typa_sql3.typa
      HAVING count(*) > 1;

REF_
----
AF
```

## *2.6 Utilisation des collections*

### 2.6.1 Nombre de vols affrétés concernant l'avion 'A1'

```
SELECT count(*) nb_vols_affretes
      FROM TABLE( SELECT  collec FROM avion_sql3 a WHERE a.na='A1' ) ;

NB_VOLS_AFFRETES
----------------
               4
```

### 2.6.2 Date et nombre de passagers transportés pour les affrètements de l'avion 'A1' dépassant les 120 passagers

```
SELECT af.date_a, af.nb_p
FROM TABLE( SELECT  collec FROM avion_sql3 a WHERE a.na='A1' ) af
WHERE af.nb_p>120;

DATE_A         NB_P
-------- ----------
11/12/00        150
```

### 2.6.3 Affrètement concernant l'avion 'A3'

```
COLUMN ref_comp_SQL3.comp FORMAT A15 HEADING ' Vols de A3';

SELECT t.ref_comp_SQL3.comp, t.date_a
      FROM TABLE(SELECT collec
             FROM avion_sql3
             WHERE na = 'A3') t;
```

```
CLEAR COLUMN

 Vols de A3     DATE_A
--------------- --------
AF              09/12/00
```

## *2.7 Utilisation des types(VALUE) et déréférencement(DEREF)*

### 2.7.1 Types des compagnies : Opérateur VALUE

```
SELECT VALUE(a) FROM avion_sql3 a WHERE a.ref_comp_sql3.comp='AF';
```

L'opérateur VALUE extrait un objet à partir d'un type.
On peut être surpris ici par l'affichage en hexadécimal des références mais Oracle affiche l'OID tel qu'il est en mémoire.

```
VALUE(A)(NA, REF_TYPA_SQL3, CAP, REF_COMP_SQL3, COLLEC(REF_COMP_SQL3, DATE_A, NB
--------------------------------------------------------------------------------
AVION_TYPE('A1', 0000220208E480552B9BFC49D8E03400306E06F2B9E445550E35C15BC3E0340
0306E06F2B9, 170, 0000220208E480552B9BF749D8E03400306E06F2B9E445550E35C05BC3E034
00306E06F2B9, AFFRETER_NESTED(ELT_AFFRETER_NESTED(0000220208E480552B9BF849D8E034
00306E06F2B9E445550E35C05BC3E03400306E06F2B9, '10/12/00', 120, 75000), ELT_AFFRE
TER_NESTED(0000220208E480552B9BF949D8E03400306E06F2B9E445550E35C05BC3E03400306E0
6F2B9, '11/12/00', 150, 87000), ELT_AFFRETER_NESTED(0000220208E480552B9BF749D8E0
3400306E06F2B9E445550E35C05BC3E03400306E06F2B9, '09/12/00', 110, 71000), ELT_AFF
RETER_NESTED(0000220208E480552B9BF749D8E03400306E06F2B9E445550E35C05BC3E03400306
E06F2B9, '19/02/01', 120, 73000)))

AVION_TYPE('A3', 0000220208E480552B9BFB49D8E03400306E06F2B9E445550E35C15BC3E0340

VALUE(A)(NA, REF_TYPA_SQL3, CAP, REF_COMP_SQL3, COLLEC(REF_COMP_SQL3, DATE_A, NB
--------------------------------------------------------------------------------
0306E06F2B9, 400, 0000220208E480552B9BF749D8E03400306E06F2B9E445550E35C05BC3E034
00306E06F2B9, AFFRETER_NESTED(ELT_AFFRETER_NESTED(0000220208E480552B9BF749D8E034
00306E06F2B9E445550E35C05BC3E03400306E06F2B9, '09/12/00', 300, 185000)))

AVION_TYPE('A4', 0000220208E480552B9BFE49D8E03400306E06F2B9E445550E35C15BC3E0340
0306E06F2B9, 120, 0000220208E480552B9BF749D8E03400306E06F2B9E445550E35C05BC3E034
00306E06F2B9, AFFRETER_NESTED())

AVION_TYPE('A6', 0000220208E480552B9BFC49D8E03400306E06F2B9E445550E35C15BC3E0340
0306E06F2B9, 160, 0000220208E480552B9BF749D8E03400306E06F2B9E445550E35C05BC3E034
00306E06F2B9, AFFRETER_NESTED())
```

### 2.7.2 Types de l'avion 'A4'

```
SELECT VALUE(a) FROM avion_sql3 a WHERE a.na='A4';

VALUE(A)(NA, REF_TYPA_SQL3, CAP, REF_COMP_SQL3, COLLEC(REF_COMP_SQL3, DATE_A, NB
--------------------------------------------------------------------------------
AVION_TYPE('A4', 0000220208E480552B9BFE49D8E03400306E06F2B9E445550E35C15BC3E0340
0306E06F2B9, 120, 0000220208E480552B9BF749D8E03400306E06F2B9E445550E35C05BC3E034
00306E06F2B9, AFFRETER_NESTED())
```

### 2.7.3 Types de la compagnie propriétaire de 'A4'

```
SELECT DEREF(a.ref_comp_sql3) FROM avion_sql3 a WHERE a.na='A4';
```

```
DEREF(A.REF_COMP_SQL3)(COMP, NOM_COMP, NB_AV)
------------------------------------------------------------------------
------
COMPAGNIE_TYPE('AF', 'Air France', 4)
```

### 2.7.4 Types des avions propriétés de la compagnie 'AF'

```
SELECT DEREF(a.ref_typa_sql3) FROM avion_sql3 a
      WHERE a.ref_comp_sql3.comp='AF';

DEREF(A.REF_TYPA_SQL3)(TYPA, NPMAX, NOMAVION, NB_AV)
------------------------------------------------------------------------
------
TYPEAVION_TYPE('A320', 180, 'Airbus A320', 2)
TYPEAVION_TYPE('B747', 400, 'Boeing Jumbo 747', 1)
TYPEAVION_TYPE('B727', 150, 'Boeing 727', 2)
TYPEAVION_TYPE('A320', 180, 'Airbus A320', 2)
```

## *2.8 Utilisation de plusieurs collections*

### 2.8.1 Affrètements (date, nombre de passagers et coût) des avions A1 et A3. Utiliser TABLE et UNION

```
SELECT affreter.date_a, affreter.nb_p, affreter.cout
      FROM avion_sql3 a, TABLE(a.collec) affreter
      WHERE a.na = 'A1'
UNION
SELECT affreter.date_a, affreter.nb_p, affreter.cout
      FROM avion_sql3 a, table (a.collec) affreter
      WHERE a.na = 'A3';

DATE_A         NB_P       COUT
-------- ---------- ----------
09/12/00        110      71000
09/12/00        300     185000
10/12/00        120      75000
11/12/00        150      87000
19/02/01        120      73000
```

### 2.8.2 Date des affrètements en commun des avions A1 et A3. Utiliser TABLE et INTERSECT

```
SELECT affreter.date_a
      FROM avion_sql3 a, TABLE(a.collec) affreter
      WHERE a.na = 'A1'
INTERSECT
SELECT affreter.date_a
      FROM avion_sql3 a, TABLE(a.collec) affreter
      WHERE a.na = 'A3';

DATE_A
--------
09/12/00
```

```
SELECT a.na, a.ref_typa_sql3.typa, affreter.date_a, affreter.nb_p,
affreter.cout
FROM avion_sql3 a, TABLE(a.collec) affreter
WHERE  a.ref_comp_sql3.comp = 'AF';

NA   REF_ DATE_A         NB_P       COUT
---- ---- -------- ---------- ----------
A1   A320 10/12/00        120      75000
A1   A320 11/12/00        150      87000
A1   A320 09/12/00        110      71000
A1   A320 19/02/01        120      73000
A3   B747 09/12/00        300     185000
```

### 2.8.3 Même question en rajoutant le code de la compagnie qui affrète chaque vol et pour les affrètements qui ne sont pas faits par la compagnie 'AF'

```
SELECT a.na, a.ref_typa_sql3.typa, affreter.date_a, affreter.nb_p,
affreter.cout, a.ref_comp_sql3.comp
FROM avion_sql3 a, TABLE(a.collec) affreter
WHERE  affreter.ref_comp_sql3.Comp <> 'AF';


NA   REF_ DATE_A         NB_P       COUT REF_
---- ---- -------- ---------- ---------- ----
A1   A320 10/12/00        120      75000 AF
A1   A320 11/12/00        150      87000 AF
```

### 2.8.4 Immatriculation des avions de la compagnie 'AF', avec le nombre d'affrètements qui ne sont pas faits par leur compagnie

```
SELECT a.na, count(affreter.ref_comp_sql3.Comp )
      FROM avion_sql3 a, TABLE(a.collec) affreter
WHERE a.ref_comp_sql3.comp ='AF' AND affreter.ref_comp_sql3.Comp <> 'AF'
GROUP BY a.na;

NA   COUNT(AFFRETER.REF_COMP_SQL3.COMP)
---- -----------------------------------
A1                                    2
```

# TP n°3 : Méthodes Héritage

## 3. Méthodes PL/SQL et Héritage

### 3.1 Méthodes PL/SQL

Il faut tout d'abord ajouter à la déclaration des types les différentes méthodes définies.

```
--Modification du type avion_type
ALTER TYPE avion_type ADD MEMBER FUNCTION nb_aff RETURN NUMBER CASCADE
/

Type modifié.

--Modification du type avion_type
ALTER TYPE avion_type ADD MEMBER FUNCTION nb_aff_2 RETURN NUMBER CASCADE
/

Type modifié.

ALTER TYPE avion_type ADD MEMBER PROCEDURE rachete_par( codecomp IN
VARCHAR ) CASCADE
/

Type modifié.
```

#### 3.1.1 Contenu du type avion_type avec les méthodes

```
CREATE OR REPLACE TYPE BODY avion_type AS

-- Renvoie le nombre d'affretements
MEMBER FUNCTION nb_aff RETURN NUMBER IS res NUMBER;
BEGIN

  SELECT COUNT(*) INTO res FROM TABLE(SELECT av.collec FROM avion_sql3 av
WHERE av.na=self.na);
  RETURN res;

END nb_aff;



MEMBER FUNCTION nb_aff_2 RETURN NUMBER IS res NUMBER;
BEGIN

        SELECT COUNT(*) INTO res FROM TABLE(SELECT av.collec FROM
      avion_sql3 av WHERE av.na=self.na) a
        WHERE a.ref_comp_sql3.comp <> (SELECT av.ref_comp_sql3.comp FROM
      avion_sql3 av WHERE av.na=SELF.na);
        RETURN res;

END nb_aff_2;


MEMBER PROCEDURE rachete_par(codecomp IN VARCHAR ) IS

  nb_comp  NUMBER;
  selfcomp VARCHAR(4);

BEGIN
```

```
      -- La compagnie n'existe pas
      SELECT count(*) INTO  nb_comp FROM compagnie_sql3 c WHERE
      comp=codecomp;

      IF nb_comp=0 THEN
            RAISE_APPLICATION_ERROR(-20001,'La compagnie n''existe pas.');
      END IF;

      -- Un avion ne peut être rachete par sa propre compagnie
      SELECT av.ref_comp_sql3.comp INTO selfcomp
            FROM avion_sql3 av
            WHERE av.na = self.na;

      IF selfcomp = codecomp THEN
            RAISE_APPLICATION_ERROR(-20002,'Impossible : cet avion
      appartient deja a cette compagnie.');
      END IF;

      UPDATE avion_sql3 a
            SET a.ref_comp_sql3 = (SELECT REF(c) FROM compagnie_sql3 c
      WHERE c.comp=codecomp)
            WHERE a.na = self.na;

END rachete_par;



MEMBER PROCEDURE ajoute_affret ( compP IN VARCHAR, date_aP IN DATE, nb_pP
In NUMBER, coutP IN NUMBER) IS
      maxi NUMBER;
      compAvion VARCHAR(3);
BEGIN

--Verifier coherence nb_p et np_max
SELECT npmax INTO maxi
            FROM typeavion_sql3 a
            WHERE a.typa =
                  (SELECT av.REF_TYPA_SQL3.typa
                        FROM avion_sql3 av WHERE av.na=self.na) ;

IF nb_pP>maxi THEN
      RAISE_APPLICATION_ERROR(-20006,'Le nombre de passagers est superieur
au nombre maximum autorise ');
END IF;

--Vérifier que la propre compagnie n'affrette pas ses propres avions
SELECT a.ref_comp_sql3.comp INTO compAvion
            FROM avion_sql3 a
            WHERE a.na = self.na;

IF compP = compAvion THEN
      RAISE_APPLICATION_ERROR(-20007,'Une compagnie ne peut pas affreter
ses propres avions.');
END IF;

--Ajout dans la collection collec de type nested table
INSERT INTO
      TABLE (SELECT av.collec FROM avion_sql3 av WHERE av.na= SELF.na)
      VALUES (elt_affreter_nested (
               (SELECT ref(c)  FROM compagnie_sql3 c WHERE c.comp=compP),
                     date_aP,
                     nb_pP,
                     coutP));
END ajoute_affret ;
```

```
END;
/

Corps de type créé.
```

### 3.1.2 Test des méthodes de compagnie_type

#### Fonction nb_aff

**a) Dans une requete**

```
SELECT a.nb_aff() FROM avion_sql3 a WHERE a.na='A1';

A.NB_AFF()
----------
         4
```

**b) Dans un bloc**

```
DECLARE
   res NUMBER;
   obj avion_type;
BEGIN
 SELECT VALUE(a) INTO obj FROM avion_sql3 a WHERE a.na='A1';
 res := obj.nb_aff();
 DBMS_OUTPUT.PUT_LINE('Resultat : '||res);
END;
/

Procédure PL/SQL terminée avec succès.

Résultat : 4
```

#### Fonction nb_aff_2 qui ne compte que les affrètements de la compagnie propriétaire

```
-Test de la procedure nb_aff_2 pour A1 et A3
DECLARE
   res NUMBER;
   obj avion_type;
BEGIN
 SELECT VALUE(a) INTO obj FROM avion_sql3 a WHERE a.na='A1';
 res := obj.nb_aff_2();
 DBMS_OUTPUT.PUT_LINE('Resultat : '||res);
END;
/
Resultat : 2

Procédure PL/SQL terminée avec succès.


DECLARE
   res NUMBER;
   obj avion_type;
BEGIN
 SELECT VALUE(a) INTO obj FROM avion_sql3 a WHERE a.na='A3';
 res := obj.nb_aff_2();
 DBMS_OUTPUT.PUT_LINE('Resultat : '||res);

END;
/

Resultat : 0

Procédure PL/SQL terminée avec succès.
```

## Méthode rachete_par

**cas 1 : compagnie inconnue**

```
DECLARE
   res NUMBER;
   obj avion_type;
BEGIN
 SELECT VALUE(a) INTO obj FROM avion_sql3 a WHERE a.na='A3';
 obj.rachete_par('TAZ');
END;
/


ERREUR à la ligne 1 :
ORA-20001: La compagnie n'existe pas.
```

**cas 2: même compagnie**

```
DECLARE
   res NUMBER;
   obj avion_type;
BEGIN
 SELECT VALUE(a) INTO obj FROM avion_sql3 a WHERE a.na='A3';
 obj.rachete_par('AF');
END;
/

ERREUR à la ligne 1 :
ORA-20002: Impossible : cet avion appartient deja a cette compagnie.
```

**cas 3 : normal**

```
DECLARE
   res NUMBER;
   obj avion_type;
BEGIN
 SELECT VALUE(a) INTO obj FROM avion_sql3 a WHERE a.na='A3';
 obj.rachete_par('AOM');

--Mise à jour du nombre d'avions
UPDATE compagnie_sql3 c
      SET c.nb_av = c.nb_av - 1
      WHERE c.comp='AF';

UPDATE compagnie_sql3 c
      SET c.nb_av = c.nb_av + 1
      WHERE c.comp='AOM';
END;
/
```

Vérification

```
SELECT * FROM compagnie_sql3;

COMP NOM_COMP                           NB_AV
---- ------------------------------ ----------
AF   Air France                             3
TAT  Transport Air Touraine                 2
AL   Air Liberté                            0
AOM  Air Outre-Mer                          1
```

```
SELECT a.ref_comp_sql3.comp FROM avion_sql3 a WHERE a.na='A3';

REF_
----
AOM
```

A3 a donc bien été racheté par AOM.

### Méthode ajoute_affret : test et trace

```
--Cas 1 : Nombre de passager supérieur au maximum autorisé

DECLARE
   obj avion_type;
BEGIN
 SELECT VALUE(a) INTO obj FROM avion_sql3 a WHERE a.na='A1';
  obj.ajoute_affret('AOM',SYSDATE,780,72000);
END;
/

ERREUR à la ligne 1 :
ORA-20006: Le nombre de passagers est superieur au nombre maximum autorise


-- Cas 2 :  même compagnie
DECLARE
   obj avion_type;
BEGIN
 SELECT VALUE(a) INTO obj FROM avion_sql3 a WHERE a.na='A6';
  obj.ajoute_affret('AF',SYSDATE,100,72000);
END;
/

ERREUR à la ligne 1 :
ORA-20007: Une compagnie ne peut pas affreter ses propres avions.


-- Cas 3 : normal
DECLARE
   obj avion_type;
BEGIN
 SELECT VALUE(a) INTO obj FROM avion_sql3 a WHERE a.na='A6';
 obj.ajoute_affret('AOM',SYSDATE,100,72000);
END;
/

Procédure PL/SQL terminée avec succès.

SELECT t.ref_comp_sql3.comp, t.date_a, t.nb_p, t.cout
       FROM TABLE (SELECT collec FROM avion_sql3 WHERE na='A6') t;

REF_ DATE_A         NB_P       COUT
---- -------- ---------- ----------
AOM  28/09/04        100      72000
```

## 3.1.3 Contenu du type compagnie_type avec les méthodes

Modification de la déclaration du type compagnie_type

```
ALTER TYPE compagnie_type
```

```
            ADD STATIC PROCEDURE insere_compagnie(compP IN VARCHAR, nomcompP IN
VARCHAR) CASCADE
```

Corps du type ompagnie_type avec les méthodes rachete_par et insere_compagnie

```
CREATE OR REPLACE TYPE BODY compagnie_type AS


      MEMBER PROCEDURE rachete_par_modif ( naP IN VARCHAR ) IS
        nb_av     NUMBER;
        selfcomp VARCHAR(4);
      BEGIN

            -- L'avion n'existe pas
            SELECT count(*) INTO  nb_av FROM avion_sql3 a WHERE a.na = naP;
            IF nb_av=0 THEN
                  RAISE_APPLICATION_ERROR(-20001,'L''avion n''existe pas.');
            END IF;

            -- Un avion ne peut être rachete par sa propre compagnie
            SELECT av.ref_comp_sql3.comp INTO selfcomp FROM avion_sql3 av WHERE
            av.na = naP;
            IF selfcomp = self.comp THEN
                  RAISE_APPLICATION_ERROR(-20002,'Impossible : cet avion
            appartient deja a cette compagnie.');
            END IF;

            -- Mise à jour dans la table avion
            UPDATE avion_sql3 a
                  SET a.ref_comp_sql3 = (SELECT REF(c) FROM compagnie_sql3 c
            WHERE c.comp=self.comp)
                  WHERE a.na = naP;

            --Mise a jour du nombre d'avions
            UPDATE compagnie_sql3 c
                  SET c.nb_av = c.nb_av + 1
                  WHERE c.comp = self.comp;

            UPDATE compagnie_sql3 c
                  SET c.nb_av = c.nb_av - 1
                  WHERE c.comp=(SELECT av.ref_comp_sql3.comp FROM avion_sql3 av
            WHERE av.na = naP);


      END rachete_par_modif;


      STATIC PROCEDURE insere_compagnie (compP  IN VARCHAR,
                                         nomcompP IN VARCHAR)   IS
      BEGIN
            INSERT INTO compagnie_sql3 VALUES (compP, nomcompP, 0);
      END insere_compagnie;

END;
/
```

### 3.1.4 Test des méthodesdu type compagnie_type

#### Méthode rachete_par _2 dans compagnie type

**cas 1 : avion inconnu**

```
DECLARE
   res NUMBER;
   obj compagnie_type;
BEGIN
 SELECT VALUE(c) INTO obj FROM compagnie_sql3 c WHERE c.comp='AF';
 obj.rachete_par_modif('A9');
END;
/

ERREUR à la ligne 1 :
ORA-20001: L'avion n'existe pas.
```

**cas 2: avion appartient deja a cette compagnie**

```
DECLARE
   res NUMBER;
   obj compagnie_type;
BEGIN
 SELECT VALUE(c) INTO obj FROM compagnie_sql3 c WHERE c.comp='AF';
 obj.rachete_par_modif('A1');
END;
/

ERREUR à la ligne 1 :
ORA-20002: Impossible : cet avion appartient deja a cette compagnie.
```

**cas 3 : normal**

```
--Situation initiale

COMP NOM_COMP                           NB_AV
---- ------------------------------ ----------
AF   Air France                              3
TAT  Transport Air Touraine                  4
AL   Air Liberté                             0
AOM  Air Outre-Mer

DECLARE
   res NUMBER;
   obj compagnie_type;
BEGIN
 SELECT VALUE(c) INTO obj FROM compagnie_sql3 c WHERE c.comp='TAT';
 obj.rachete_par_modif('A1');
END;
/

Procédure PL/SQL terminée avec succès.



SELECT * FROM compagnie_sql3;

COMP NOM_COMP                           NB_AV
---- ------------------------------ ----------
AF   Air France                              5
TAT  Transport Air Touraine                  2
AL   Air Liberté                             0
AOM  Air Outre-Mer                           1

-- A1 a bien été racheté par TAT :
SELECT a.ref_comp_sql3.comp FROM avion_sql3 a WHERE a.na='A1';

REF_
----
TAT
```

### Méthode insere_compagnie dans compagnie type

**Situation initiale :**

```
SELECT * FROM compagnie_sql3;

COMP NOM_COMP                           NB_AV
---- ------------------------------ ----------
AF   Air France                              3
TAT  Transport Air Touraine                 4
AL   Air Liberté                             0
AOM  Air Outre-Mer                           1
TA   TunisAir                                0
```

**Test :**

```
BEGIN
     compagnie_type.insere_compagnie('TA','TunisAir');
END;
/

Procédure PL/SQL terminée avec succès.
```

**Situation finale :**

```
SELECT * FROM compagnie_sql3;

COMP NOM_COMP                           NB_AV
---- ------------------------------ ----------
AF   Air France                              3
TAT  Transport Air Touraine                  4
AL   Air Liberté                             0
AOM  Air Outre-Mer                           1
TA   TunisAir                                0
```

## *3.2 Modification de type*

### 3.2.1 Ajout de la colonne Tonnage

```
ALTER TYPE typeavion_type ADD ATTRIBUTE tonnage NUMBER(4)
/

Type modifié
```

Vérification :

```
DESC typeavion_type
Nom                                       NULL ?   Type
----------------------------------------- -------- ----------------------------
TYPA                                                VARCHAR2(4)
NPMAX                                               NUMBER(3)
NOMAVION                                            VARCHAR2(30)
NB_AV                                               NUMBER(3)
TONNAGE                                            NUMBER(4)
```

### 3.2.2 Mise à jour de la colonne tonnage

```
UPDATE typeavion_sql3 SET tonnage = 250 WHERE typa='B747';
UPDATE typeavion_sql3 SET tonnage = 120 WHERE typa='A320';
UPDATE typeavion_sql3 SET tonnage = 170 WHERE typa='A310';
UPDATE typeavion_sql3 SET tonnage = 110 WHERE typa='B727';
UPDATE typeavion_sql3 SET tonnage = 90 WHERE typa='CRJ1';
```

### 3.2.3 Ajout de l'avion A99 qui n'a pas encore d'affretements

```
INSERT INTO avion_sql3 VALUES('A99',
        ( SELECT REF(a) FROM typeavion_sql3 a WHERE a.typa='A320'),
                    140,
                    (SELECT REF(c) FROM compagnie_sql3 c WHERE
                                                        c.comp='AOM'),
                     affreter_nested());
```

```
Insertion de l'affretement

DECLARE
    var_obj avion_type;
BEGIN
            SELECT VALUE(av) INTO var_obj FROM avion_sql3 av
             WHERE av.na='A2';
            var_obj.ajoute_affret('AOM', SYSDATE,90, 30000);
END;
/
```

```
SELECT c.ref_comp_sql3.comp, c.date_a, c.nb_p, c.cout FROM TABLE( SELECT
collec FROM avion_sql3 a WHERE a.na='A2') c;

REF_ DATE_A         NB_P       COUT
---- -------- ---------- ----------
AOM  28/09/04         90      30000
```

## *3.3 Héritage*

### 3.3.1 Création des nouveaux types et tables

```
DROP TABLE Comp_Transport_SQL3;
DROP TABLE Comp_Fret_SQL3;
DROP TYPE Comp_Transport_type;
DROP TYPE Comp_Fret_type;
DROP TYPE Comp_virtuelle_type;
DROP TYPE Societe_type;
DROP TYPE Adr_type;
```

```
CREATE OR REPLACE TYPE Adr_type AS OBJECT
(        nRue     NUMBER,
       rue  CHAR(50),
       ville      CHAR(20))
       NOT FINAL
       INSTANTIABLE
/
Type créé.
```

```
CREATE OR REPLACE TYPE Societe_type AS OBJECT
 (    nSiret      NUMBER,
      adresse adr_type,
      budget NUMBER,
      nb_employe NUMBER,
      NOT FINAL MEMBER PROCEDURE affiche)
      NOT FINAL
      NOT INSTANTIABLE
/

Type créé.

CREATE OR REPLACE TYPE Comp_virtuelle_type UNDER Societe_type
(       comp VARCHAR(4),
      nom_comp VARCHAR(30),
      nb_av NUMBER,
      FINAL STATIC PROCEDURE listecomp)
      NOT FINAL
      NOT INSTANTIABLE
/

Type créé.

CREATE OR REPLACE TYPE Comp_Transport_type UNDER Comp_virtuelle_type
      (nb_pax_total NUMBER,
       prix_moy_billet NUMBER,
       FINAL MEMBER PROCEDURE compte_pax,
       FINAL MEMBER PROCEDURE prix_moy,
       OVERRIDING FINAL MEMBER PROCEDURE affiche)
      FINAL
      INSTANTIABLE
/

Type créé.

CREATE OR REPLACE TYPE Comp_Fret_type UNDER Comp_virtuelle_type
      (tonnage NUMBER,
       aeroport_base VARCHAR(30),
       MEMBER PROCEDURE calcul_tonnage,
       OVERRIDING FINAL MEMBER PROCEDURE affiche)
      FINAL
      INSTANTIABLE
/

Type créé.

CREATE TABLE Comp_Fret_SQL3 OF Comp_fret_type
 (     CONSTRAINTS pk_comp_fret PRIMARY KEY (comp));

Table créée.

CREATE TABLE Comp_Transport_SQL3 OF Comp_Transport_type
      (CONSTRAINTS pk_comp_trans PRIMARY KEY (comp));

Table créée.
```

## 3.3.2. Organisation des données

```
--Compagnie de transport

INSERT INTO Comp_Transport_sql3 (comp, nom_comp)
      SELECT comp, nom_comp FROM compagnie_sql3
            WHERE comp='AF' OR  comp='TAT' ;
```

```
--Compagnie de fret

INSERT INTO Comp_fret_sql3 (comp, nom_comp) SELECT comp, nom_comp FROM
compagnie_sql3  WHERE comp!='AF' AND  comp!='TAT' ;
```

**-- Vérification**

```
SELECT comp, nom_comp FROM Comp_Transport_sql3;

COMP NOM_COMP
---- ------------------------------
TAT  Transport Air Touraine
AF   Air France


SELECT comp, nom_comp FROM Comp_fret_sql3;

COMP NOM_COMP
---- ------------------------------
AL   Air Liberté
AOM  Air Outre-Mer
TA   TunisAir
```

## Pour la compagnie de transport renseigner les colonnes issues du type societe_type

```
UPDATE comp_transport_sql3
            SET nSiret=1,
            adresse=(Adr_type(15,'Place du Capitole','TOULOUSE')),
            budget=150,
            nb_employe=1000
      WHERE comp='AF';

1 ligne mise à jour.

UPDATE Comp_fret_sql3  SET
            nSiret=2,adresse=(Adr_type(54,'Place de l''etoile','PARIS')),
            budget=200,
            nb_employe=2000
       WHERE comp='TA';


1 ligne mise à jour.
```

Vérification :

```
SELECT comp, nom_comp, nSiret, budget, nb_employe FROM
Comp_transport_sql3;

COMP NOM_COMP                          NSIRET     BUDGET NB_EMPLOYE
---- ------------------------------ ---------- ---------- ----------
TAT  Transport Air Touraine
AF   Air France                             1        150       1000

SELECT comp, nom_comp, nSiret, budget, nb_employe FROM Comp_fret_sql3;

COMP NOM_COMP                          NSIRET     BUDGET NB_EMPLOYE
---- ------------------------------ ---------- ---------- ----------
AL   Air Liberté
```

```
AOM  Air Outre-Mer
TA   TunisAir                                2        200       2000
```

### 3.3.3 Mise à jour des pointeurs de la table avion_sql3

```
ALTER TYPE avion_type ADD ATTRIBUTE tran REF Comp_Transport_type CASCADE
/

Type modifié.

ALTER TYPE avion_type ADD ATTRIBUTE fret REF Comp_fret_type CASCADE
/

Type modifié.
```

#### *Compagnie de transport*

```
UPDATE avion_sql3 a
      SET a.tran = (SELECT REF(c) FROM comp_transport_sql3 c WHERE
comp='AF' )
      WHERE a.ref_comp_sql3.comp='AF' ;

1 ligne(s) mise(s) à jour.

UPDATE avion_sql3 a
      SET a.tran = (SELECT REF(c) FROM comp_transport_sql3 c WHERE
comp='TAT' )
      WHERE a.ref_comp_sql3.comp='TAT' ;

4 ligne(s) mise(s) à jour.
```

#### *Compagnie de fret*

```
UPDATE avion_sql3 a
      SET a.fret = (SELECT REF(c) FROM comp_fret_sql3 c WHERE comp='AOM' )
      WHERE a.ref_comp_sql3.comp='AOM' ;

2 ligne(s) mise(s) à jour.
```

Vérification :

```
SELECT na, a.tran.comp, a.fret.comp , a.ref_comp_sql3.comp FROM avion_sql3
a;

NA   TRAN FRET REF_
---- ---- ---- ----
A1   TAT       TAT
A2   TAT       TAT
A3        AOM  AOM
A4   TAT       TAT
A5   TAT       TAT
A6   AF        AF
A99       AOM  AOM

7 ligne(s) sélectionnée(s).


ALTER TYPE elt_affreter_nested ADD ATTRIBUTE tran REF Comp_Transport_type
CASCADE
/

Type modifié.
```

```
ALTER TYPE elt_affreter_nested ADD ATTRIBUTE fret REF Comp_fret_type
CASCADE
/

Type modifié.
```

### *Affrètement d'avion*

```
Modification de la référence à la main

UPDATE TABLE (
   SELECT collec FROM avion_sql3 avi WHERE avi.ref_comp_sql3.comp ='AF') a
   SET a.tran
      = (SELECT REF(c) FROM comp_transport_sql3 c WHERE c.comp = 'AF');

1 ligne mise à jour.


UPDATE TABLE (
SELECT collec FROM avion_sql3 avi WHERE avi.ref_comp_sql3.comp ='AF') a
SET a.tran = (SELECT REF(c) FROM comp_transport_sql3 c WHERE c.comp =
'AF');

1 ligne mise à jour.


UPDATE TABLE(SELECT collec FROM avion_sql3 avi WHERE
avi.ref_comp_sql3.comp ='AOM') a
      SET a.fret = (SELECT REF(c) FROM comp_fret_sql3 c WHERE c.comp =
'AOM');

1 ligne mise à jour.

AL et TA n'ont pas affreté, donc pas besoin de les modifier


Vérification pour A1 :

SELECT av.na, a.tran
      FROM TABLE(SELECT collec FROM avion_sql3 co
                  WHERE co.ref_comp_sql3.comp= 'AF') a, avion_sql3 av
      WHERE av.na='A1';

NA
----
TRAN
--------------------------------------------------------------------------
A1
0000220208E528E932BC2A584CE03400306E06F2B9E526D740FA1F4BD6E03400306E06F2B9
```

## 3.3.4 Destruction de la table compagnie

```
DROP TABLE compagnie_sql3;
```

## 3.3.5 Programmation des méthodes affiche(), listecomp(), calcul_tonnage(), ...

```
ALTER TYPE societe_type ADD MEMBER PROCEDURE affiche
/
```

```
ALTER TYPE comp_fret_type ADD MEMBER PROCEDURE affiche
/
ALTER TYPE comp_transport_type  ADD overriding  MEMBER PROCEDURE affiche
/
ALTER TYPE comp_transport_type  ADD MEMBER PROCEDURE prix_moyen
/
ALTER TYPE comp_transport_type  ADD MEMBER PROCEDURE compte_pax
/
ALTER TYPE comp_fret_type ADD MEMBER PROCEDURE calcul_tonnage
/
```

-- Corps du type **societe_type**

```
CREATE OR REPLACE TYPE BODY societe_type AS

      MEMBER PROCEDURE affiche IS
      BEGIN

        DBMS_OUTPUT.PUT_LINE('Societe ' || self.nsiret || ' au capital de ' ||
self.budget || ' euros.');
            DBMS_OUTPUT.PUT_LINE('Composé de ' || self.nb_employe || '
employés.');
            DBMS_OUTPUT.PUT_LINE('Adresse : '|| self.adresse.nrue||' '||
self.adresse.rue||' '|| self.adresse.ville );

      END affiche;
END;
/
```

--Corps du type **comp_transport**

```
CREATE OR REPLACE TYPE BODY comp_transport_type AS

      overriding MEMBER PROCEDURE affiche IS
      BEGIN
            DBMS_OUTPUT.PUT_LINE('La compagnie ' || self.nom_comp || ' ' ||
self.comp || ' possède ' || self.nb_av || 'avion.');
            DBMS_OUTPUT.PUT_LINE('Nb passagers transporté : ' ||
self.nb_pax_total || '.');
            DBMS_OUTPUT.PUT_LINE('Prix moyen d''un billet : ' ||
self.prix_moy_billet || '.');
      END affiche;


      MEMBER PROCEDURE compte_pax IS
            nb_pax_tot NUMBER;
      BEGIN
            SELECT SUM(c.nb_p) INTO nb_pax_tot FROM avion_sql3 a, TABLE
(a.collec) c
                  WHERE a.tran.comp=SELF.comp;

            UPDATE comp_transport_sql3 c
                  SET nb_pax_total = nb_pax_tot
                  WHERE c.comp = SELF.comp;

      END compte_pax;


      MEMBER PROCEDURE prix_moyen ( compAfretP IN VARCHAR, date_aP  IN DATE) IS

            nb_pV NUMBER;
            coutV NUMBER;
      BEGIN
            SELECT c.nb_p INTO nb_pV, c.cout into coutV
```

```
                  FROM TABLE(SELECT collec FROM avion_sql3 WHERE
a.tran.comp=self.comp )       c
                        WHERE c.tran.comp = compAfretP
                        AND   c.date_a    = date_aP;

            UPDATE comp_transport_type c
                  SET prix_moy_billet = (coutV / nb_pV)
                  WHERE c.comp = compAfretP;

      END prix_moyen;

END;
/
```

```
--corps du type comp_fret
CREATE OR REPLACE TYPE BODY comp_fret_type AS

      overriding MEMBER PROCEDURE affiche IS
      BEGIN
            DBMS_OUTPUT.PUT_LINE('La compagnie ' || self.nom_comp || ' ' ||
self.comp || ' possède ' || self.nb_av || 'avion.');
            DBMS_OUTPUT.PUT_LINE('Tonnage : ' || self.tonnage || '.');
            DBMS_OUTPUT.PUT_LINE('Basé à ' || self.aeroport_base || '.');
      END affiche;



      MEMBER FUNCTION calcul_tonnage RETURN number IS
            tonn  number;
      BEGIN
            SELECT SUM(a.ref_typa_sql3.tonnage) INTO tonn
                  FROM avion_sql3 a
                  WHERE a.fret.comp = SELF.comp;

            RETURN tonn;

      END calcul_tonnage;

END;
/
```

```
--Corps du type comp_virtuelle_type
CREATE OR REPLACE TYPE BODY comp_virtuelle_type AS

      MEMBER FUNCTION listecomp RETURN number IS
      BEGIN
            DBMS_OUTPUT.PUT_LINE('Compagnie ' || self.comp || ' );
            DBMS_OUTPUT.PUT_LINE('Nom : ' || self.nom_comp|| '.');
            DBMS_OUTPUT.PUT_LINE('Nombre d''avions :  ' || self.nb_av || '.');
      END listecomp;

END;
/
```

### 3.3.6 Test des méthodes

#### Méthode Affiche

```
-- Afifchage d'une compagnie de transport
DECLARE
      obj comp_transport_type ;
BEGIN
```

```
      SELECT VALUE(c) INTO obj FROM comp_transport_sql3 c WHERE
c.comp='AF';
      obj.affiche();
END;
/

-- Afifchage d'une compagnie de fret
DECLARE
      obj comp_fret_type ;
BEGIN
      SELECT VALUE(c) INTO obj FROM comp_fret_sql3 c WHERE c.comp='AOM';
      obj.affiche();
END;
/


-- Appel de la fonction calcul du tonnage
DECLARE
      obj comp_fret_type ;
BEGIN
      SELECT VALUE(c) INTO obj FROM comp_fret_sql3 c WHERE c.comp='AOM';
      obj.calcul_tonnage();
END;
/



-- Fonction compte_pax()
DECLARE
      nb_pax_tot NUMBER;
BEGIN
      SELECT SUM(c.nb_p) INTO nb_pax_tot FROM avion_sql3 a, TABLE
(a.collec) c
            WHERE a.tran.comp='AF';

      UPDATE comp_transport_sql3 c
            SET nb_pax_total = nb_pax_tot
            WHERE c.comp = 'AF';
END;
/

SELECT c.nb_pax_total FROM comp_transport_sql3 c WHERE c.comp='AF';
```