



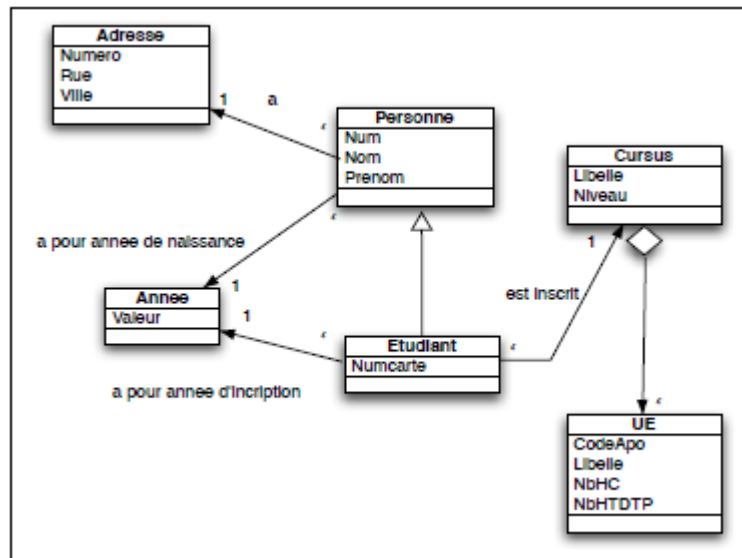
## **Module : BD OBJET RELATIONNELLES TP2**

Préparé par : ELMOURJANI Zineb

-

Année universitaire : 2020/2019

Soit le diagramme de classes UML simple ci-dessous.



## Création de types et de tables objet utilisant ces types

- 1- Créer les types correspondants aux classes Année (Année T), Adresse (Adresse T), Personne (Personne T).

```
create type adresse_t as object (
    numero number,
    rue number,
    ville varchar2(50));

create type annee_t as object(
    valeur number);

create type personne_t as object(
    num number,
    nom varchar2(50),
    prenom varchar2(50),
    annee_naissance annee_t,
    adresse adresse_t
);
```

Sortie de script x

Tâche terminée en 0,06 secondes

Elément Type ADRESSE\_T compilé

Elément Type ANNEE\_T compilé

Elément Type PERSONNE\_T compilé

Vérifier la description de ces types (desc) et vérifier les descriptions au sein des tables de la metabase USER TYPES et USER TYPE ATTRS.

```

-- select * from user_types;
-- drop type adresse_t force;

```

Sortie de script x

Résultat de requête x

Toutes les lignes extraites : 9 en 0,005 secondes

| TYPE_NAME                     | TYPE_OID                         | TYPECODE   | ATTRIBUTES | METHODS | PREDEFINED | INCOMPLETE |
|-------------------------------|----------------------------------|------------|------------|---------|------------|------------|
| 1 REPCAT\$_OBJECT_NULL_VECTOR | 296EAC6057114D26B123A36D2679C516 | OBJECT     | 4          | 0 NO    | NO         |            |
| 2 DEPARTEMENT_TYPE            | BEAA6416B8FA4EC0B23A732BCD6DE642 | OBJECT     | 3          | 0 NO    | NO         |            |
| 3 EMPLOYE_TYPE                | 98AF367638E644709B56AF3E4A0993A2 | OBJECT     | 6          | 0 NO    | NO         |            |
| 4 PRENOMS                     | 231DEC6C64874C66BF8453CB32035E6C | COLLECTION | 0          | 0 NO    | NO         |            |
| 5 TRAVAILLEUR_TYPE            | EC7F3FFCDE674C089D9333EB16DA135B | OBJECT     | 3          | 0 NO    | NO         |            |
| 6 NOM                         | 3180E5F3E81A428CBC6FC1CA24BE6CAB | OBJECT     | 2          | 0 NO    | NO         |            |
| 7 ADRESSE_T                   | 3B04A0E968AB462F8328795A7C7D6019 | OBJECT     | 3          | 0 NO    | NO         |            |
| 8 ANNEE_T                     | 7A5872B599304C4BA3D63BD6AA9FD4D2 | OBJECT     | 1          | 0 NO    | NO         |            |
| 9 PERSONNE_T                  | 2B0E1CBA12E4BB4A9AC0B92693B46AF  | OBJECT     | 5          | 0 NO    | NO         |            |

```

select * from user_type_attrs;
drop type adresse_t force;
drop type annee_t force;

```

| TYPE_NAME                      | ATTR_NAME       | ATTR_TYPE_MOD | ATTR_TYPE_OWNER | ATTR_TYPE_NAME   |
|--------------------------------|-----------------|---------------|-----------------|------------------|
| 1 PERSONNE_T                   | NUM             | (null)        | (null)          | NUMBER           |
| 2 ANNEE_T                      | VALEUR          | (null)        | (null)          | NUMBER           |
| 3 ADRESSE_T                    | RUE             | (null)        | (null)          | NUMBER           |
| 4 ADRESSE_T                    | NUMERO          | (null)        | (null)          | NUMBER           |
| 5 NOM                          | D               | (null)        | (null)          | NUMBER           |
| 6 TRAVAILLEUR_TYPE             | SALAIRE         | (null)        | (null)          | NUMBER           |
| 7 EMPLOYE_TYPE                 | SALAIRE         | (null)        | (null)          | NUMBER           |
| 8 DEPARTEMENT_TYPE             | NUM             | (null)        | (null)          | NUMBER           |
| 9 REPCAT\$_OBJECT_NULL_VECTOR  | NULL_VECTOR     | (null)        | (null)          | RAW              |
| 10 REPCAT\$_OBJECT_NULL_VECTOR | TYPE_HASHCODE   | (null)        | (null)          | RAW              |
| 11 PERSONNE_T                  | PRENOM          | (null)        | (null)          | VARCHAR2         |
| 12 PERSONNE_T                  | NOM             | (null)        | (null)          | VARCHAR2         |
| 13 ADRESSE_T                   | VILLE           | (null)        | (null)          | VARCHAR2         |
| 14 NOM                         | N               | (null)        | (null)          | VARCHAR2         |
| 15 TRAVAILLEUR_TYPE            | NOM             | (null)        | (null)          | VARCHAR2         |
| 16 EMPLOYE_TYPE                | NOM             | (null)        | (null)          | VARCHAR2         |
| 17 EMPLOYE_TYPE                | MATRICULE       | (null)        | (null)          | VARCHAR2         |
| 18 DEPARTEMENT_TYPE            | LIEU            | (null)        | (null)          | VARCHAR2         |
| 19 DEPARTEMENT_TYPE            | NOM             | (null)        | (null)          | VARCHAR2         |
| 20 REPCAT\$_OBJECT_NULL_VECTOR | TYPE_NAME       | (null)        | (null)          | VARCHAR2         |
| 21 REPCAT\$_OBJECT_NULL_VECTOR | TYPE_OWNER      | (null)        | (null)          | VARCHAR2         |
| 22 EMPLOYE_TYPE                | DEPARTEMENT     | REF           | SYSTEM          | DEPARTEMENT_TYPE |
| 23 EMPLOYE_TYPE                | SUPERIEUR       | REF           | SYSTEM          | EMPLOYE_TYPE     |
| 24 TRAVAILLEUR_TYPE            | PRENOMS_ARRAY   | (null)        | SYSTEM          | PRENOMS          |
| 25 PERSONNE_T                  | ADRESSE         | (null)        | SYSTEM          | ADRESSE_T        |
| 26 PERSONNE_T                  | ANNEE_NAISSANCE | (null)        | SYSTEM          | ANNEE_T          |

- 2- Créer la table objet Personnes définie sur le type Personne T en ajoutant les contraintes suivantes :
  1. la clé primaire est définie sur NUM,
  2. le nom doit être UNIQUE
  3. l'année de naissance doit être comprise entre 1900 et 2007

```

create table personnes of personne_t (primary key (num), unique(nom) );
ALTER TABLE personnes ADD CONSTRAINT annee CHECK(annee_naissance.valeur between 1900 and 2007)

```

Sortie de script x | Tâche terminée en 0,316 secondes

Table PERSONNES créé(e).

Table PERSONNES modifié(e).

Vérifier en consultant USER OBJECT TABLES et USER Constraints.

```
select * from user_object_tables;
```

| TABLE_NAME     | TABLESPACE_NAME | CLUSTER_NAME | IOT_NAME | STATUS | PCT_FREE | PCT_USED | INI_TRANS | MAX_TRANS | INITIAL_EXTENT | NEXT_EXTENT |
|----------------|-----------------|--------------|----------|--------|----------|----------|-----------|-----------|----------------|-------------|
| 1 PERSONNES    | SYSTEM          | (null)       | (null)   | VALID  | 10       | 40       | 1         | 255       | 65536          | (null)      |
| 2 DEPARTEMENTS | SYSTEM          | (null)       | (null)   | VALID  | 10       | 40       | 1         | 255       | 65536          | (null)      |
| 3 EMPLOYES     | SYSTEM          | (null)       | (null)   | VALID  | 10       | 40       | 1         | 255       | 65536          | (null)      |

```
select * from user_constraints where CONSTRAINT_NAME='ANNEE';
```

| OWNER    | CONSTRAINT_NAME | CONSTRAINT_TYPE | TABLE_NAME | SEARCH_CONDITION                                    | R_OWNER | R_CONSTRAINT_NAME | DELETE_RULE | STATUS  | DEFERRABLE     | DEFERRED  | VALIDATOR |
|----------|-----------------|-----------------|------------|---|---------|-------------------|-------------|---------|----------------|-----------|-----------|
| 1 SYSTEM | ANNEE           | C               | PERSONNES  | annee_naissance.valeur between 1900 and 2007 (null) | (null)  | (null)            | (null)      | ENABLED | NOT DEFERRABLE | IMMEDIATE | VALIDATOR |

- 3- Insérer quelques objets dans la table Personnes (habitant Montpellier, Narbonne, Beziers, Nimes), vérifier que les contraintes sont opérantes.

```
insert into personnes VALUES (1, 'ahmad', 'ahmad', annee_t(2001), adresse_t(1,1, 'Montpellier'));
insert into personnes VALUES (2, 'samir', 'samir', annee_t(2002), adresse_t(2,2, 'Narbonne'));
insert into personnes VALUES (3, 'kamal', 'kamal', annee_t(2003), adresse_t(3,3, 'Beziers'));
insert into personnes VALUES (4, 'nour', 'nour', annee_t(2004), adresse_t(4,4, 'Nimes'));
insert into personnes VALUES (5, 'nour', 'mary', annee_t(1000), adresse_t(5,5, 'Montpellier'));
```

Sortie de script x

Tâche terminée en 0,123 secondes

1 ligne inséré.

1 ligne inséré.

1 ligne inséré.

1 ligne inséré.


Erreur commençant à la ligne: 59 de la commande -  
insert into personnes VALUES (5, 'nour', 'mary', annee\_t(1000), adresse\_t(5,5, 'Montpellier'))  
Rapport d'erreur -  
ORA-02290: check constraint (SYSTEM.ANNEE) violated

#### 4- Interrogations

1. Donner la totalité des informations de toutes les personnes

```
SQL> select value(p) from personnes p;  
  
VALUE(P){NUM, NOM, PRENOM, ANNEE_NAISSANCE(VALEUR), ADRESSE(NUMERO, RUE, VILLE))  
-----  
PERSONNE_T(1, 'ahmad', 'ahmad', ANNEE_T(2001), ADRESSE_T(1, 1, 'Montpellier'))  
PERSONNE_T(2, 'samir', 'samir', ANNEE_T(2002), ADRESSE_T(2, 2, 'Narbonne'))  
PERSONNE_T(3, 'kamal', 'kamal', ANNEE_T(2003), ADRESSE_T(3, 3, 'Beziers'))  
PERSONNE_T(4, 'nour', 'nour', ANNEE_T(2004), ADRESSE_T(4, 4, 'Nimes'))  
  
SQL> |
```

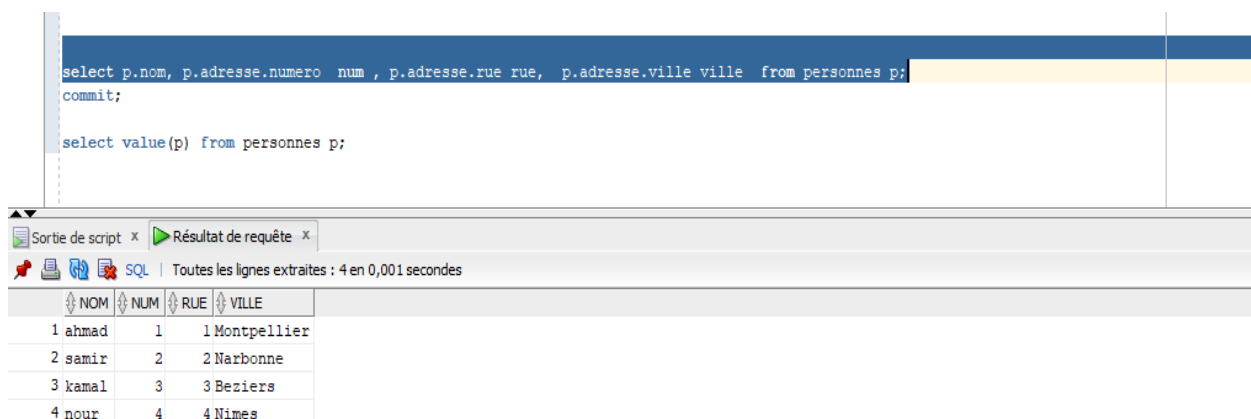
2. Donner le nom et l'adresse de toutes les personnes

 sqlplus

```
SQL> select nom, adresse from personnes;
```

```
NOM  
-----  
ADRESSE(NUMERO, RUE, VILLE)  
-----  
ahmad  
ADRESSE_T(1, 1, 'Montpellier')  
  
samir  
ADRESSE_T(2, 2, 'Narbonne')  
  
kamal  
ADRESSE_T(3, 3, 'Beziers')  
  
NOM  
-----  
ADRESSE(NUMERO, RUE, VILLE)  
-----  
nour  
ADRESSE_T(4, 4, 'Nimes')  
  
SQL> |
```

 sqldeveloper



The screenshot shows the SQL Developer interface. The top pane contains the following SQL code:

```
select p.nom, p.adresse.numero num , p.adresse.rue rue, p.adresse.ville ville from personnes p;  
commit;  
  
select value(p) from personnes p;
```

The bottom pane shows the results of the first query in a table format:

|   | NOM   | NUM | RUE | VILLE       |
|---|-------|-----|-----|-------------|
| 1 | ahmad | 1   |     | Montpellier |
| 2 | samir | 2   |     | Narbonne    |
| 3 | kamal | 3   |     | Beziers     |
| 4 | nour  | 4   |     | Nimes       |

### 3. Donner les références de toutes les personnes

```
SQL> select ref(p) from personnes p;

REF(P)
-----
00002802098C590965A83F4076BFF419F410026AE293DD067F891C48D8BD882DA88B88E0840040EA
FA0000

0000280209BE7F81CA954C4172882CE6F7989B6C0093DD067F891C48D8BD882DA88B88E0840040EA
FA0001

0000280209BECAD00F405F4DD59C7A7BA28564455A93DD067F891C48D8BD882DA88B88E0840040EA
FA0002

000028020938F65D0574E647ECAE998C4D248CEE1093DD067F891C48D8BD882DA88B88E0840040EA
FA0003

REF(P)
-----
```

### 4. Donner le nom et l'année de naissance des personnes résidant dans la ville de Nimes

```
SQL> select nom , p.annee_naissance.valeur from personnes p where p.adresse.ville='Nimes';

NOM                                ANNEE_NAISSANCE.VALEUR
-----
nour                                2004

SQL> |
```

## Les collections :

D'après le modèle un cursus est composé de plusieurs UE.

1. Créer le type correspondant à la classe UE (UE T). Vérifier la description de ces types (desc) et vérifier les descriptions au sein des tables de la metabase USER TYPES et USER TYPE ATTRS.

```
create type ue_t as object(
  codeApo number,
  libelle varchar2(50),
  nbHC number,
  nbHTDTP number
);
```

Sortie de script x Résultat de requête x

Tâche terminée en 0,104 secondes

Validation (commit) terminée.

Elément Type UE\_T compilé

```
SQL> desc ue_t;
Name                                Null?                                Type
-----
CODEAPO                             NUMBER
LIBELLE                             VARCHAR2(50)
NBHC                                 NUMBER
NBHTDTP                             NUMBER

SQL> select * from user_types where type_name='UE_T';

TYPE_NAME                           TYPE_OID
-----
TYPECODE                             ATTRIBUTES  METHODS  PRE  INC  FIN  INS
SUPERTYPE_OWNER                      SUPERTYPE_NAME                                LOCAL_ATTRIBUTES
LOCAL_METHODS  TYPEID
UE_T
OBJECT                                DA87591F9D5F402FAD9900ACB8E22973
                                4                                0 NO NO YES YES
```

2. . Pour représenter la collection d'UE d'un cursus, deux solutions sont possibles correspondant aux types collections possibles dans Oracle : les collections illimitées qui seront utilisées comme tables imbriquées (nested tables) et les tableaux predimensionnés (varray).

```
create type ue_varray is varray(20) of ue_t;  
create type ue_nested as table of ue_t;
```

Sortie de script x Résultat de requête x  
Tâche terminée en 0,227 secondes

Validation (commit) terminée.

Élément Type UE\_T compilé

Validation (commit) terminée.

Élément Type UE\_VARRAY compilé

3. Créer ensuite des tables Cursus relationnelles utilisant ces collections (en prévoyant les contraintes de clés primaires).

```
create table cursus_varray (  
  num number,  
  libelle varchar2(50),  
  niveau varchar2(50),  
  ue ue_varray);
```

Sortie de script x  
Tâche terminée en 0,056 secondes

Table CURSUS\_VARRAY supprimé(e).

Table CURSUS\_VARRAY créé(e).

```
create table cursus_nested (  
  num number,  
  libelle varchar2(30),  
  niveau varchar2(30),  
  ue ue_nested  
  ) nested table ue store as ue_tab;
```

Sortie de script x  
Tâche terminée en 0,087 secondes

Table CURSUS\_NESTED créé(e).

- Créer divers types Coursus T définis en utilisant ces collections, ainsi que des tables Coursus objets (en prévoyant les contraintes de clés primaires).

```
create type cursus_type_varray as object(  
  num number,  
  libelle varchar2(30),  
  niveau varchar2(30),  
  ue ue_varray  
);
```

Sortie de script x

Tâche terminée en 0,066 secondes

Elément Type CURSUS\_TYPE\_VARRAY compilé

```
create table cursusV of cursus_type_varray(primary key (num));
```

Sortie de script x

Tâche terminée en 0,062 secondes

Table CURSUSV créé(e).

- Insérer des cursus et leur collection d'UE.

```
insert into cursusV values(1,'GEGM','tronc commun',ue_varray(ue_t(1,'MP',13,14),ue_t(3,'TT',77,99)));  
  
insert into cursusV values(2,'MIP','tronc commun',ue_varray(ue_t(2,'AM',13,14),ue_t(2,'BB',77,99)));
```

Sortie de script x

Tâche terminée en 0,08 secondes

1 ligne inséré.

1 ligne inséré.



## 6. Interrogations (à réaliser dans les divers cas de figures)

### 1. Donner le libellé, le niveau et la collection d'Ue de chaque cursus

```
select libelle ,niveau ,cursor(select * from table(ue)) from cursusV ;
```

Résultat de requête x

Toutes les lignes extraites : 2 en 0,086 secondes

|   | LIBELLE | NIVEAU       | CURSUS(SELECT * FROM TABLE(UE))  |
|---|---------|--------------|--|
| 1 | GEGM    | tronc commun | {<CODEAPO=1, LIBELLE=MP, NBHC=13, NBHTDTP=14>, <CODEAPO=3, LIBELLE=IT, NBHC=77, NBHTDTP=99>, } |
| 2 | MIP     | tronc commun | {<CODEAPO=2, LIBELLE=AM, NBHC=13, NBHTDTP=14>, <CODEAPO=2, LIBELLE=BB, NBHC=77, NBHTDTP=99>, } |

### 2. Donner les codes Apogée des UE du cursus de tronc commun de libellé GEGM

```
select u.codeApo from the (select ue from cursusV where niveau='tronc commun' and libelle='GEGM') u;
```

Résultat de requête x

Toutes les lignes extraites : 2 en 0,038 secondes

|   | CODEAPO |
|---|---------|
| 1 | 1       |
| 2 | 3       |

### 3. Donner les libellés des cursus et les codes Apogée des UE ayant un nombre d'heures de cours > 15 heures

```
select r.libelle, u.codeApo from cursusV r, the (select ue from cursusV where r.num=num) u where u.nbHC>5;
```

Résultat de requête x

Toutes les lignes extraites : 4 en 0,003 secondes

|   | LIBELLE | CODEAPO |
|---|---------|---------|
| 1 | GEGM    | 1       |
| 2 | GEGM    | 3       |
| 3 | MIP     | 2       |
| 4 | MIP     | 2       |

# La spécialisation et les références

## 1. Créer le type Etudiant T

The screenshot shows two SQL scripts in a development environment. The first script modifies the `PERSONNE_T` type to be non-final and cascade changes. The second script creates a new type `ETUDIANT_T` under `PERSONNE_T`, adding attributes `numCarte` (number) and `cursus` (reference to `CURSUS_TYPE_VARRAY`). Both scripts are executed successfully, as indicated by the status bars.

```
alter type personne_t not final cascade;
```

Sortie de script x  
Tâche terminée en 0,957 secondes  
Type PERSONNE\_T modifié(e).

```
create type etudiant_t under personne_t (numCarte number, cursus ref cursus_type_varray );
```

Sortie de script x  
Tâche terminée en 0,146 secondes  
Elément Type ETUDIANT\_T compilé

## 2. Créer une table objet Etudiants

The screenshot shows a SQL script creating the `ETUDIANTS` table using the `ETUDIANT_T` type. The `num` attribute is set as the primary key. The script is executed successfully.

```
2- create table etudiants of etudiant_t(primary key (num));
```

Sortie de script x  
Tâche terminée en 0,838 secondes  
Table ETUDIANTS créé(e).

3. Réaliser des insertions dans cette table. On peut commencer à insérer des valeurs à des valeurs de la table Personnes et initialiser le cursus a la valeur NULL. Ensuite pour un étudiant donné on peut faire une mise à jour de la table Etudiants en lui affectant un des cursus préalablement créés (via la référence objet correspondante).

The screenshot shows two SQL scripts. The first script inserts a new student record into the `ETUDIANTS` table. The second script updates the `cursus` attribute of the student with `num=5` to reference the `GEGM` curriculum. Both scripts are executed successfully.

```
3- insert into etudiants values(5,'elmourjani','zineb',annee_t(1992),adresse_t(2,300,'Montpellier'),3000,null);  
  
update etudiants set cursus = (Select ref(c) from cursusV c where libelle='GEGM') where num=5;
```

Sortie de script x  
Tâche terminée en 0,049 secondes  
1 ligne inséré.  
1 ligne mis à jour.

4. Interrogations diverses (quelques exemples ... puis laisser votre imagination s'exercer)
1. Nom, Année d'inscription et cursus des étudiants

```
select e.nom, e.annee_naissance.valeur, e.cursus.libelle from etudiants e;
```

Résultat de requête x

Toutes les lignes extraites : 1 en 0,077 secondes

| NOM          | ANNEE_NAISSANCE.VALEUR | CURSUS.LIBELLE |
|--------------|------------------------|----------------|
| 1 elmourjani | 1992                   | GEGM           |

2. Nom, Code Apogées des UE suivies par un étudiant donné

```
update etudiants set cursus = (Select ref(c) from cursusV c where libelle='GEGM') where num=5;
```

Résultat de requête x

Toutes les lignes extraites : 2 en 0,066 secondes

| LIBELLE | CODEAPO |
|---------|---------|
| 1 MP    | 1       |
| 2 TT    | 3       |