Bases de Données Avancées

TD/TP : SQL3

– USTHB Master 01 IL–
M. AZZOUZ
Dernière mis à jour :Avril 2020

Exercice 01

- **LDD:** langage de définition de données (types par défaut, types définis par utilisateur, collection, héritage, transformation de modèle objet(diagramme de classes) en un modèle relationnel objet).
- **LMD:** langage de manipulation de données (insertion, mise à jour)
- **LID:** langage d'interrogation de données(des requêtes simples: usage de des opérateurs value, value is of, treat et table).

Classe Personne

Personne
-NSS -Nom - Nom de famille - Prénom -Date_naiss -Adresse - Rue - Numéro - Ville - Pays - Code postal -Téléphone (1,6)

En analysant la classe personne on a :

-**Nom** et **adresse** : sont des attributs composés nécessitent la définition d'un nouveau type structuré.

- **Téléphone**: une collection de 6 éléments en maximum, pour celui là on a le choix entre un type varray ou une table imbriquée.

-**Les autres attributs** sont de type simple qui existent par défaut en oracle

Classe Personne

Description de type tnom:

Personne
-NSS -Nom - Nom de famille - Prénom -Date_naiss -Adresse - Rue - Numéro - Ville - Pays - Code postal -Téléphone (1,6)

Attribut Nom lui correspond un type structuré comme suit:

create type tnom as **object** (nomdefamille varchar2(50), prenom varchar2(50));

/

-Pour définir un type structuré on utilise la clause **as object**.

```
SQL> create type tnom as object (nomdefamille varchar2(50), prenom varchar2(50));
  2  /

Type crÚÚ.

SQL> desc tnom
 Nom                                       NULL ?   Type
 ----------------------------------------- -------- ----------------------------
 NOMDEFAMILLE                                       VARCHAR2(50)
 PRENOM                                             VARCHAR2(50)
```

Classe Personne

Description de type tadresse:

Personne
-NSS -Nom - Nom de famille - Prénom -Date_naiss -Adresse - Rue - Numéro - Ville - Pays - Code postal -Téléphone (1,6)

Attribut Adresse lui correspond un type structuré comme suit:

```
create type tadresse as object (rue varchar2(50), numero integer, ville varchar2(50), pays varchar2(50), codepostal varchar2(5));
/
```

```
SQL> create type tadresse as object(rue varchar2(50), numero integer, ville varchar2(50), pays varchar2(50), codepostal varchar2(5));
  2  /

Type crÚÚ.

SQL> desc tadresse
 Nom                                       NULL ?   Type
 ----------------------------------------- -------- ----------------------------
 RUE                                                VARCHAR2(50)
 NUMERO                                             NUMBER(38)
 VILLE                                              VARCHAR2(50)
 PAYS                                               VARCHAR2(50)
 CODEPOSTAL                                         VARCHAR2(5)
```

Classe Personne

Description de type ttelephone:

Personne
-NSS -Nom - Nom de famille - Prénom -Date_naiss -Adresse - Rue - Numéro - Ville - Pays - Code postal -Téléphone (1,6)

Attribut Téléphone lui correspond un type collection fini comme suit:

```
create type ttelephone as varray(6) of varchar2(12);
/
```

varray indique qu'il s'agit d'une collection d'éléments finis, pour plus de flexibilité il vaut mieux utiliser les collections de type table imbriquée (nested table) pour ne pas faire référence à des fonctions précises aux tableaux. Les fonctions sont manipulées dans des programmes PL/SQL.

```
SQL> create type ttelephone as varray(6) of varchar2(12);
  2  /

Type crÚÚ.

SQL> desc ttelephone
 ttelephone VARRAY(6) OF VARCHAR2(12)
```

Classe Personne

Description de type tpersonne:

Personne
-NSS -Nom - Nom de famille - Prénom -Date_naiss -Adresse - Rue - Numéro - Ville - Pays - Code postal -Téléphone (1,6)

create type tpersonne as **object** (nss integer, nom tnom, date_naiss date, adresse tadresse, telephone ttelephone) **Not Final**;
/

- Pour une classe on définit un type structuré on utilise la clause **as object**.
- La clause **Not Final** indique que le type peut avoir des sous types

```
SQL> create type tpersonne as object (nss integer, nom tnom, date_naiss date, adresse tadresse, telephone ttelephone) Not Final;
  2  /

Type crÚÚ.

SQL> desc tpersonne
 tpersonne est NOT FINAL
 Nom                                       NULL ?   Type
 ----------------------------------------- -------- ----------------------------
 NSS                                                NUMBER(38)
 NOM                                                TNOM
 DATE_NAISS                                         DATE
 ADRESSE                                            TADRESSE
 TELEPHONE                                          TTELEPHONE
```

Classe Personne

Ajout d'une méthode au type tpersonne:

Ajouter une méthode pour le type tpersonne se fait en deux étapes: la première la définition de la signature de la méthode et la deuxième phase est la déclaration de corps de la méthode "**body**", la mention **member** indique la méthode s'applique sur des objets.

- Signature: alter type tpersonne *add member function* **calcul_age** return numeric cascade;

- Body: create or replace type body tpersonne
as member function calcul_age return numeric
is
age_val number;
Begin
Select round((sysdate-**self.**date_naiss)/365) into age_val From dual;
Return age_val;
End ;
End;
/

Classe Personne

Ajout de la table personne:

create table personne of **tpersonne** (primary key(nss));

- La table **personne** c'est une table **objet** car elle est définie sur un type objet tpersonne, chaque ligne de la table a un identifiant objet **OID** par défaut mais pour avoir une intégrité on définit la clé primaire, en Orienté objet on peut avoir deux objets avec les mêmes valeurs et le système leur donne un OID différent, OID est géré par le SE.

```
SQL> create table personne of tpersonne (primary key(nss));

Table crÚÚe.

SQL> desc personne
 Nom                                       NULL ?   Type
 ----------------------------------------- -------- ----------------------------
 NSS                                       NOT NULL NUMBER(38)
 NOM                                                TNOM
 DATE_NAISS                                         DATE
 ADRESSE                                            TADRESSE
 TELEPHONE                                          TTELEPHONE
```

Table Personne

Insertion dans la table personne:

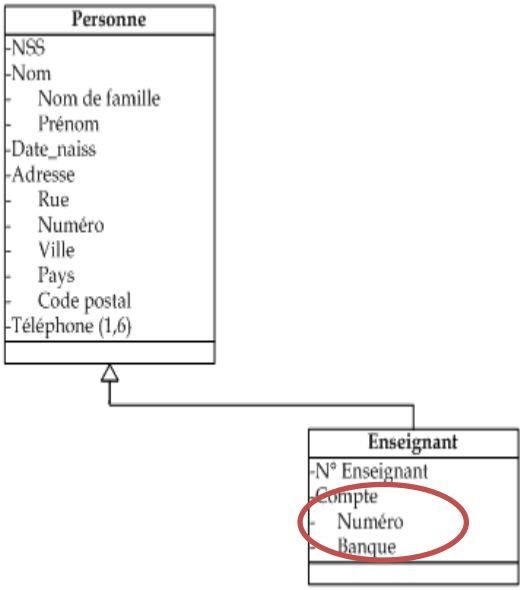
insert into personne values(**tpersonne**('123456789', tnom('ADIMI', 'Mohamed'), '01/01/1980', tadresse('rue de la gare', '22','Alger', 'Algérie', '1600'), ttelephone('023112345', '0554321921')));

- Pour insérer une ligne dans la table on fait référence au constructeur de type « tpersonne ». Pour les types définis par utilisateur comme **nom**, **adresse** et **telephone**, on introuduit la valeur en indiquant le nom de constructeur qui le même que le nom de type.

```
SQL> insert into personne values(tpersonne('123456789', tnom('ADIMI', 'Mohamed'), '01/01/1980', tadresse('rue de la gare', '22','Alger', 'Algérie', '1600'), ttelephon
e('023112345', '0554321921')));

1 ligne crÚÚe.
```

Classe Personne

Description de sous type tenseignant:



- Un enseignant a un attribut structuré **compte**, il faut définir son type.

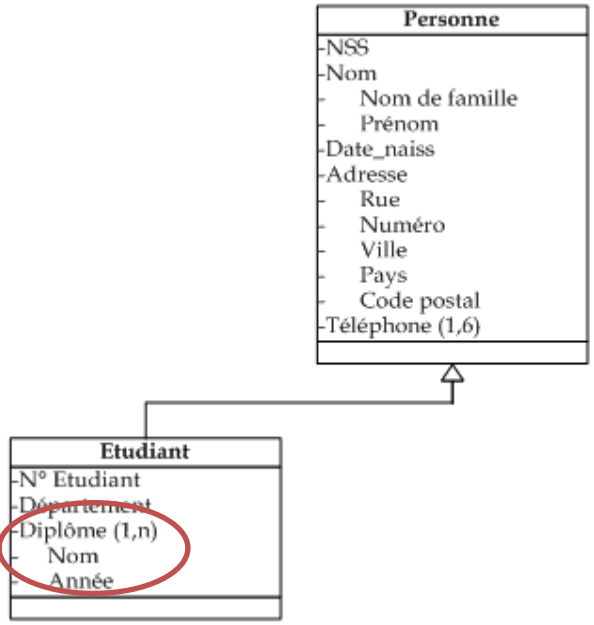
create type **tcompte** as object (numero integer, banque varchar2(50));
/
create type tenseignant **under** tpersonne (numero_ens integer, compte tcompte);
/

- La clause **under** permet de définir des sous types

```
SQL> create type  tcompte as object (numero integer, banque varchar2(50));
  2  /

Type crÚÚ.

SQL> desc tcompte
 Nom                                                                                                 NULL ?   Type
 --------------------------------------------------------------------------------------------------- -------- ------------------------------
 NUMERO                                                                                                       NUMBER(38)
 BANQUE                                                                                                       VARCHAR2(50)

SQL> create type tenseignant under tpersonne (numero_ens integer, compte tcompte);
  2  /

Type crÚÚ.

SQL> desc tenseignant
 tenseignant constitue une extension de BDA1920.TPERSONNE
 Nom                                                                                                 NULL ?   Type
 --------------------------------------------------------------------------------------------------- -------- ------------------------------
 NSS                                                                                                          NUMBER(38)
 NOM                                                                                                          TNOM
 DATE_NAISS                                                                                                   DATE
 ADRESSE                                                                                                      TADRESSE
 TELEPHONE                                                                                                    TTELEPHONE
 NUMERO_ENS                                                                                                   NUMBER(38)
 COMPTE                                                                                                       TCOMPTE
```

Classe Personne

Description de sous type tetudiant: (1)



Un étudiant a un ensemble de diplômes, il s'agit d'une collection→ table imbriquée des diplômes.

1. Définir le type d'un élément (un diplôme)

```
create type tdiplome as object(nom varchar2(50), annee integer);
/
```

2. Utiliser le type tdiplome pour définir le type collection de diplômes

```
create type t_set_diplomes as table of tdiplome;
/
```

3. Utiliser le type collection t_set_diplomes dans la définition de type tetudiant

```
create type tetudiant under tpersonne (num_etud integer, departement varchar2(50), diplome t_set_diplomes);
/
```

Classe Personne

Description de sous type tetudiant: (2)

```
SQL> create type tdiplome as object(nom varchar2(50), annee integer);
  2  /

Type crÚÚ.

SQL> desc tdiplome
 Nom                                                                                     NULL ?   Type
 --------------------------------------------------------------------------------------- -------- ----------------------
 NOM                                                                                              VARCHAR2(50)
 ANNEE                                                                                            NUMBER(38)

SQL> create type t_set_diplomes as table of tdiplome;
  2  /

Type crÚÚ.

SQL> desc t_set_diplomes
 t_set_diplomes TABLE OF TDIPLOME
 Nom                                                                                     NULL ?   Type
 --------------------------------------------------------------------------------------- -------- ----------------------
 NOM                                                                                              VARCHAR2(50)
 ANNEE                                                                                            NUMBER(38)

SQL> create type tetudiant under tpersonne (num_etud integer, departement varchar2(50), diplome t_set_diplomes);
  2  /

Type crÚÚ.

SQL> desc tetudiant
 tetudiant constitue une extension de BDA1920.TPERSONNE
 Nom                                                                                     NULL ?   Type
 --------------------------------------------------------------------------------------- -------- ----------------------
 NSS                                                                                              NUMBER(38)
 NOM                                                                                              TNOM
 DATE_NAISS                                                                                       DATE
 ADRESSE                                                                                          TADRESSE
 TELEPHONE                                                                                        TTELEPHONE
 NUM_ETUD                                                                                         NUMBER(38)
 DEPARTEMENT                                                                                      VARCHAR2(50)
 DIPLOME                                                                                          T_SET_DIPLOMES
```

Classe Personne

Insertion dans les sous classes:

-Pour les deux sous classes on ne crée pas de table, on garde que la table de super type Personne, la gestion des types est prise lors de l'insertion en spécifiant à chaque fois le constructeur.

-insert into **personne** values (tetudiant('123123123',tnom('MERABETI','Adam'),'01/05/1985',tadresse('Dido uche Mourad','','Alger','Algérie',''), ttelephone(),'999','Informatique', t_set_diplomes()));

-insert into **personne** values (tenseignant('666999666',tnom('LAMARI','Meriem'),'04/06/1975',tadresse('Boule vard Colonel Amirouche','99','Alger','Algérie',''), ttelephone(),'777',tcompte('310123456789','BDA')));

-insert into **personne** values (tenseignant('556978566',tnom('SALEMI','Ahmed'),' 04/06/1965',tadresse('Ismail Yafsah','99','Alger','Algérie',''), ttelephone(),'787',tcompte('330123489756','BEA')));

Classe Personne

Upadate: (1)

-Modifier les numéros de téléphones de la personne, en utilisant la clause update, dans le cas ou la personne n'a pas de numéros de téléphone pas de problème mais dans le cas contraire, on aura un écrasement des anciens numéros de téléphones, pour éviter ça il faut soit déclarer le type comme une table imbriquée et la mise à jour se fait en utilisant la clause **update table** ou bien garder le type varray et faire la mise à jour par un programme PL/SQL dans lequel vous aller faire un parcourt de tableau le plus normale possible en utilisant des fonctions spéciales au varray.

Classe Personne

Upadate: (2)

-Mise à jour avec une valeur connue:

```
update personne
set telephone=ttelephone('022342222','066543333')
where nom=tnom('MERABETI','Adam');
```

- Mise à jour avec une valeur récupérée à partir d'une requête

```
update personne
set telephone=(select telephone from personne where nom=tnom('MERABETI','Adam'))
where nom=tnom('LAMARI','Meriem');
```

Classe Personne

Requêtes sur la table personne:

- **Afficher les personnes qui sont des étudiants:**

```
select p.nom.nomdefamille, p.nom.prenom,
treat(value(p) as tetudiant).num_etud as numero
from personne p
where value(p) is of (tetudiant);
```

```
SQL> select p.nom.nomdefamille, p.nom.prenom, treat(value(p) as tetudiant).num_etud as numero
  2  from personne p
  3  where value(p) is of (tetudiant);

NOM.NOMDEFAMILLE                                   NOM.PRENOM                                              NUMERO
-------------------------------------------------- -------------------------------------------------- ----------
MERABETI                                           Adam                                                      999
```

-La fonction value(X) is of typeY: elle rend un booléen : true dans le cas positif (X est de typeY) et false dans le cas négative(X n'est pas de typeY).

- La fonction treat(value(X) as typeY): vaut permet de faire un transtypage bas, passer de l'objet super type (tpersonne) vers son détail (le sous type), avec cette fonction on accède au détails des sous types (les autres attributs et méthodes).

Classe Personne

Requêtes sur la table personne:

- **Afficher les personnes qui sont des enseignants:**

```
select p.nom.nomdefamille, p.nom.prenom,
treat(value(p) as tenseignant).numero_ens as numero
from personne p
where value(p) is of (tenseignant);
```

```
SQL> select p.nom.nomdefamille, p.nom.prenom, treat(value(p) as tenseignant).numero_ens as numero
  2  from personne p
  3  where value(p) is of (tenseignant);

NOM.NOMDEFAMILLE                                   NOM.PRENOM                                            NUMERO
-------------------------------------------------- -------------------------------------------------- ----------
LAMARI                                             Meriem                                                    777
SALEMI                                             Ahmed                                                     787
```

- **Afficher lez personnes qui ne sont ni des enseignants, ni des étudiants:**

```
select p.nom.nomdefamille, p.nom.prenom
from personne p
where value(p) is of (only tpersonne);
```

```
SQL> select p.nom.nomdefamille, p.nom.prenom
  2  from personne p
  3  where value(p) is of (only tpersonne);

NOM.NOMDEFAMILLE                                   NOM.PRENOM
-------------------------------------------------- --------------------
ADIMI                                              Mohamed
```

Classe Personne

Requêtes sur la table personne:

-on cherche les couples de personnes qui partagent au moins un numéro de téléphone:(1)

-D'abord faisons un aperçu sur la table personne:

```
select distinct p.nss, t.column_value
from personne p, table(p.telephone) t
order by p.nss;
```

```
       NSS COLUMN_VALUE
---------- ------------
 123123123 022342222
 123123123 066543333
 123456789 023112345
 123456789 0554321921
 666999666 022342222
 666999666 066543333

6 ligne(s) sélectionnée(s).
```

Ces deux personnes partagent un numéro de téléphone

-Téléphone est une collection, pour récupérer les éléments de la collection on le fait en utilisant l'opérateur **table** qui permet d'aplatir ou désimbriquer la collection.

column_value: car l'ensemble des téléphones ne possède pas de noms on la défini sur un latérale varchar (create type ttelephone as varray(6) of varchar2(12);/)

Classe Personne

Requêtes sur la table personne:

-on cherche les couples de personnes qui partagent au moins un numéro de téléphone:(2)

```
select distinct p1.nom.nomdefamille, p1.nom.prenom,
p2.nom.nomdefamille, p2.nom.prenom
from personne p1, table(p1.telephone) t1, personne p2,
table(p2.telephone) t2
where p1.nss<p2.nss
and t1.column_value=t2.column_value;
```

-Le résultat de la requête c'est un couple de personne→ la table personne apparait deux fois dans le select les alias p1 et p2.

-Les personnes recherchées partagent au moins un numéro de téléphone d'où l'utilisation de table pour récupérer les téléphones de chaque personne. Les alias t1: table(p1.telephone) et t2: table(p2.telephone) représentent les téléphones des personnes p1 et p2.

Classe Personne

Requêtes sur la table personne:

-on cherche les couples de personnes qui partagent au moins un numéro de téléphone:(3)

```
select distinct p1.nom.nomdefamille, p1.nom.prenom,
p2.nom.nomdefamille, p2.nom.prenom
from personne p1, table(p1.telephone) t1, personne p2,
table(p2.telephone) t2
where p1.nss<p2.nss
and t1.column_value=t2.column_value;
```

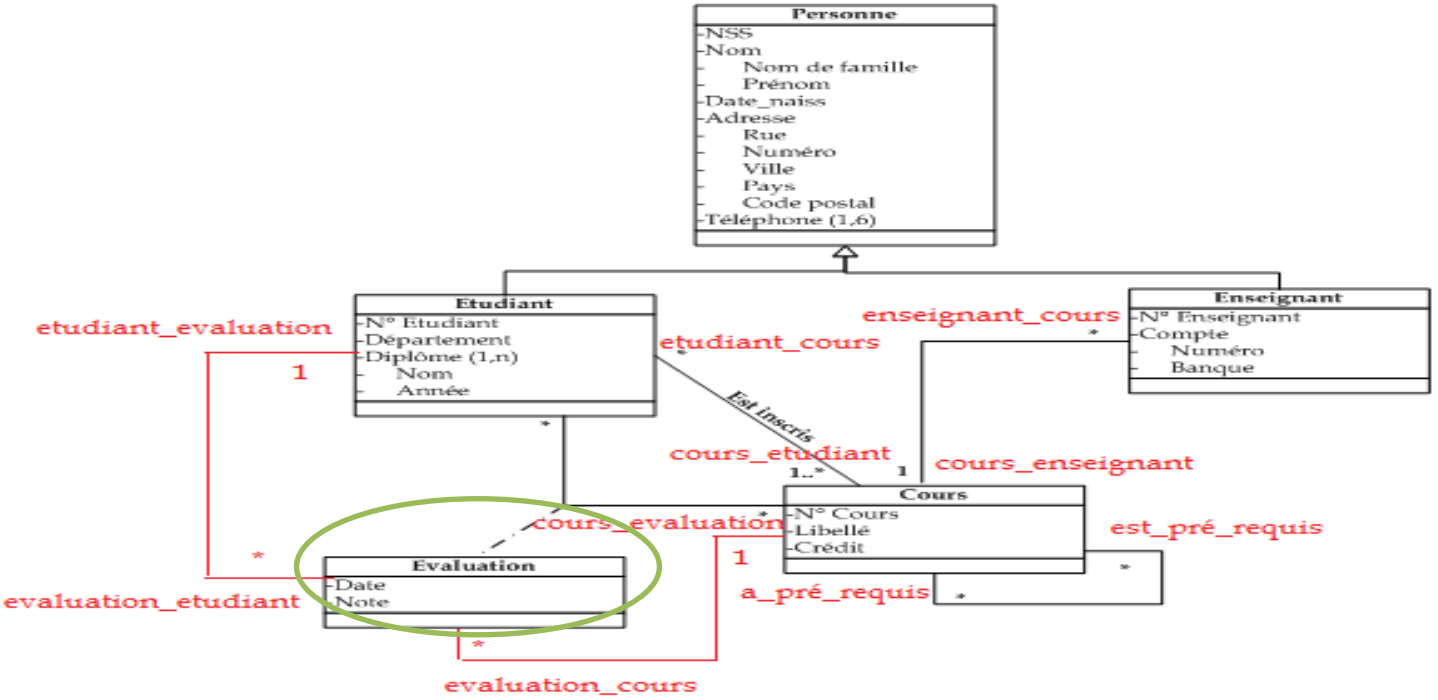
-La condition where p1.nss<p2.nss pour prendre la personne une fois dans le test (where) une façon d'ordonner les personnes.

-La condition and t1.column_value=t2.column_value pour tester l'existence d'un numéro de téléphone commun

-La clause distinct dans le select permet d'éliminer les doublons car on peut avoir des personnes qui partagent plusieurs numéros de téléphones.

Association en SQL3

Transformation des associations:

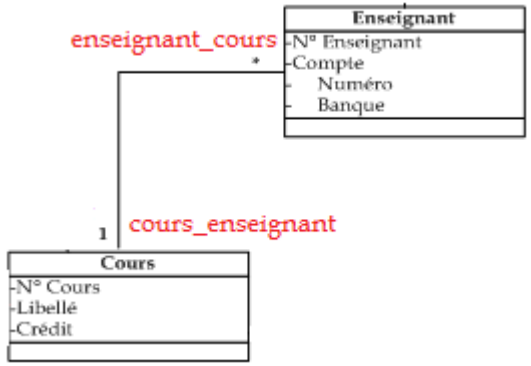


- Pour chaque association il faut définir les deux rôles. Par exemple l'association entre enseignant et cours on a les deux rôles(enseignant_cours et cours_enseignant).
- Toute association à classe d'association il faut la transformer en deux associations regarder le cas de l'association évaluation.

Association en SQL3

Stockage des associations:(1)

1. Association père-fils 1..*:



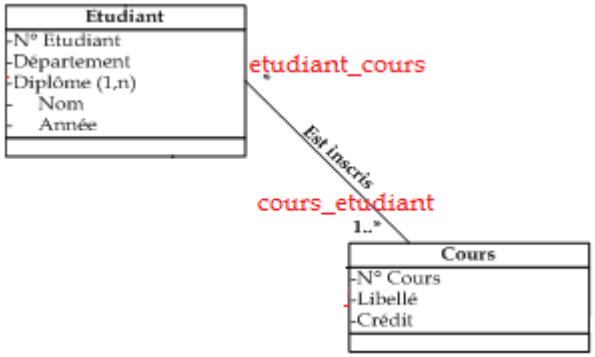
-Pour la classe père (cours) on ajoute un attribut qui sauvegarde les références des fils(enseignants) qui ont enseigné ce cours d'où cours_enseignant représente une table imbriquée des références des enseignants.

- pour la classe fils(enseignant) on ajout un attribut qui sauvegarde la référence de père(cours) enseigné par cet enseignant d'où enseignant_cours représente la référence vers le cours enseigné.

Association en SQL3

Stockage des associations:(2)

2. Association plusieurs à plusieurs *..*:



-Pour chaque la classe étudiant on ajoute un attribut qui sauvegarde les références des cours ou l'étudiant est inscrit d'où etudiant_cours représente une table imbriquée des références des cours.

- pour la classe cours on ajout un attribut qui sauvegarde les références des étudiants inscrits dans cet cours d'où cours_etudiant représente une table imbriquée des références des étudiants.

Association en SQL3

Définition des types cours et évaluation ainsi que les associations:(1)

-Les types incomplets: vu que les types s'utilisent de manière bidirectionnelle pour définir les association. Définir un type incomplet est la manière d'indiquer au SGBD que le type existe sans rentrer dans le détail juste définir le nom de type.

```
create type tcours;
/
create type tevaluation;
/
```

```
SQL> create type tcours;
  2  /

Type crÚÚ.
SQL> create type tevaluation;
  2  /

Type crÚÚ.
```

- Les deux types tcours et tevaluation sont des types incomplets.

Association en SQL3

Définition des types cours et évaluation ainsi que les associations:(2)

-Mise à jour de type tenseignant: (1)

un enseignant enseigne un seul cours sur le schéma, pour chaque enseignant il faut garder la référence de cours enseigné, la clause **ref** permet de définir des attributs de type référence.

alter type tenseignant add attribute enseignant_cours ref tcours cascade;

•Alter type permet de modifier le schéma d'un type.

•Add attribute pour ajouter un attribut au type et cascade permet de forcer la mise à jour.

•La clause ref indique que l'attribut est de type référence.

Association en SQL3

Définition des types cours et évaluation ainsi que les associations:(3)

-Mise à jour de type tenseignant: (2)

```
SQL> alter type tenseignant add attribute enseignant_cours ref tcours cascade;

Type modifiÚ.

SQL> desc tenseignant
 tenseignant constitue une extension de BDA1920.TPERSONNE
 Nom                                                                                   NULL ?   Type
 ------------------------------------------------------------------------------------- -------- ----------------
 NSS                                                                                            NUMBER(38)
 NOM                                                                                            TNOM
 DATE_NAISS                                                                                     DATE
 ADRESSE                                                                                        TADRESSE
 TELEPHONE                                                                                      TTELEPHONE
 NUMERO_ENS                                                                                     NUMBER(38)
 COMPTE                                                                                         TCOMPTE
 ENSEIGNANT_COURS                                                                               REF OF TCOURS

METHOD
------
 MEMBER FUNCTION CALCUL_AGE RETURNS NUMBER
```

Association en SQL3

Définition des types cours et évaluation ainsi que les associations:(4)

-Mise à jour de type tetudiant: (1)

•Sur le schéma précédent un étudiant à le rôle etudiant_cours qui sauvegarde les références des cours ou il est déjà inscrit et le rôle etudiant_evaluation qui sauvegarde les références de ses évaluations.

•etudiant_cours→ **collection** de références des objets de type tcours.

```
create type t_set_ref_cours as table of ref tcours;
/
alter type tetudiant add attribute etudiant_cours t_set_ref_cours cascade;
```

•etudiant_evaluation→ **collection** de références des objets de type tevaluation.

```
create type t_set_ref_evaluation as table of ref tevaluation;
/
alter type tetudiant add attribute etudiant_evaluation t_set_ref_evaluation cascade;
```

Association en SQL3

Définition des types cours et évaluation ainsi que les associations:(5)

-Mise à jour de type tetudiant: (2)

```
SQL> create type t_set_ref_cours as table of ref tcours;
  2  /

Type crÚÚ.

SQL> create type t_set_ref_evaluation as table of ref tevaluation;
  2  /

Type crÚÚ.

SQL> alter type tetudiant add attribute etudiant_cours t_set_ref_cours cascade;

Type modifiÚ.

SQL> alter type tetudiant add attribute etudiant_evaluation t_set_ref_evaluation cascade;

Type modifiÚ.

SQL> desc tetudiant
 tetudiant constitue une extension de BDA1920.TPERSONNE
 Nom                                                                                   NULL ?   Type
 ------------------------------------------------------------------------------------- -------- ----------------------
 NSS                                                                                            NUMBER(38)
 NOM                                                                                            TNOM
 DATE_NAISS                                                                                     DATE
 ADRESSE                                                                                        TADRESSE
 TELEPHONE                                                                                      TTELEPHONE
 NUM_ETUD                                                                                       NUMBER(38)
 DEPARTEMENT                                                                                    VARCHAR2(50)
 DIPLOME                                                                                        T_SET_DIPLOMES
 ETUDIANT_COURS                                                                                 T_SET_REF_COURS
 ETUDIANT_EVALUATION                                                                            T_SET_REF_EVALUATION

METHOD
------
 MEMBER FUNCTION CALCUL_AGE RETURNS NUMBER
```

Association en SQL3

Définition des types cours et évaluation ainsi que les associations:(6)

-Mise à jour de type incomplet tcours: (1)

Sur le schéma précédent un cours à le rôle cours_etudiant qui sauvegarde les références des étudiants inscrits, le rôle cours_evaluation qui sauvegarde les références des évaluations , le rôle cours_enseignant qui sauvegarde les références des enseignants qui enseignent ce cours, le rôle a_pre_requis qui sauvegarde les références des ses cours prés requis et le rôle est_pre_requis qui sauvegarde les références des cours ou il est défini comme pré requis.

Association en SQL3

Définition des types cours et évaluation ainsi que les associations:(7)

-Mise à jour de type incomplet tcours: (2)

•Cours_etudiants→ **collection** de références des objets de type tetudiant.

```
create type t_set_ref_etudiant as table of ref tetudiant;
/
```

•Cours_enseignant→ **collection** de références des objets de type tenseignant.

```
create type t_set_ref_enseignant as table of ref tenseignant;
/
```

•Cours_evaluation→ **collection** de références des objets de type tevaluation. Ce type est déjà défini pour la mise à jour de type tetudiant (create type t_set_ref_evaluation as table of ref tevaluation;/).

•a_pre_requis ou est_pre_requis → **collection** de références des objets de type tcours. Ce type est déjà défini pour la mise à jour de type tetudiant (create type t_set_ref_cours as table of ref tcours;/).

Association en SQL3

Définition des types cours et évaluation ainsi que les associations:(8)

-Mise à jour de type incomplet tcours: (3)

```
SQL> create or replace type tcours as object (numero_cours varchar2(5), libelle varchar2(50), credit integer,  est_pre_requis t_set_ref_cours, a_pre_requis t_set_ref_
cours, cours_enseignant t_set_ref_enseignant , cours_evaluation t_set_ref_evaluation, cours_etudiant t_set_ref_etudiant);
  2  /

Type crÚÚ.
```

```
SQL> desc tcours
 Nom                                                                     NULL ?   Type
 ----------------------------------------------------------------------- -------- ------------------------
 NUMERO_COURS                                                                     VARCHAR2(5)
 LIBELLE                                                                          VARCHAR2(50)
 CREDIT                                                                           NUMBER(38)
 EST_PRE_REQUIS                                                                   T_SET_REF_COURS
 A_PRE_REQUIS                                                                     T_SET_REF_COURS
 COURS_ENSEIGNANT                                                                 T_SET_REF_ENSEIGNANT
 COURS_EVALUATION                                                                 T_SET_REF_EVALUATION
 COURS_ETUDIANT                                                                   T_SET_REF_ETUDIANT
```

Rôles dans les différentes associations

Association en SQL3

Définition des types cours et évaluation ainsi que les associations:(9)

-Mise à jour de type incomplet tevaluation:

Sur le schéma précédent une évalution à le rôle evaluation_etudiant qui sauvegarde la références de l'étudiant concerné par cette évaluation et le rôle evaluation_cours qui sauvegarde la référence de cours concerné par cette évaluation.

create or replace type tevaluation as object (evaluation_cours ref tcours, evaluation_etudiant ref tetudiant, date_evaluation date, note number(4,2));

/

```
SQL> create or replace type tevaluation as object (evaluation_cours ref tcours, evaluation_etudiant ref tetudiant, date_evaluation date, note number(4,2));
  2  /

Type crÚÚ.

SQL> desc tevaluation
 Nom                                                                                     NULL ?   Type
 --------------------------------------------------------------------------------------- -------- ------------------------------------------------------------
 EVALUATION_COURS                                                                                 REF OF TCOURS
 EVALUATION_ETUDIANT                                                                              REF OF TETUDIANT
 DATE_EVALUATION                                                                                  DATE
 NOTE                                                                                             NUMBER(4,2)
```

Association en SQL3

Définition des tables cours et évaluation:(10)

-Table cours:

```
create  table cours of tcours (PRIMARY KEY(numero_cours))
nested table est_pre_requis store as table_est_pre_requis,
nested table a_pre_requis store as table_a_pre_requis,
nested table cours_etudiant store as table_cours_etudiant,
nested table cours_enseignant store as table_cours_enseignant,
nested table cours_evaluation store as table_cours_evaluation;
```

NB: pour chaque attribut qui de type table imbriquée (nested table) il faut lui définir son store (stockage physique).

-Table evaluation:

```
create table evaluation of tevaluation (foreign key(evaluation_cours) references cours, foreign key (evaluation_etudiant) references personne);
```

Association en SQL3

Tables imbriquée stockage logique et physique

Logiquement chaque ligne possède sa propre table imbriquée mais physiquement le SGBD gère une seule zone de stockage (store). Cette zone est gérée automatiquement par le SGBD en utilisant des pointeurs.

Table cours

NUMERO_COURS	LIBELLE	CREDIT	…	COURS_ENSEIGNANT	…
BD	Bases de données	5	…	Adresse de début dans le store de la table imbriqué de la ligne BD @1	…
SI	Système d'information	4	…	Adresse de début dans le store de la table imbriqué de la ligne SI @2	…
BDA	Base de données avancées	5	…	Adresse de début dans le store de la table imbriqué de la ligne BDA @3	…

Store (stockage) de la table imbriquée COURS_ENSEIGNANT

Référence de l'enseignant	
Ref(enseignant1)	@1
Ref(enseignant2)	
Ref(enseignant3)	
Ref(enseignant1)	@2
Ref(enseignant3)	
Ref(enseignant4)	
Ref(enseignant4)	@3
Ref(enseignant5)	

Exercice 02

```
create type T_Personne ;
/
create type T_SET_Tag as table of
Varchar2(30);
/
create type T_Message as object (
texte varchar2(500),
dateEcrit Date,
tags T_SET_Tag
);
/
create type T_SET_Message as
table of T_Message;
/
```

```
create type T_Contact as object(
p ref T_Personne,
depuis Date
)
/
create type T_SET_Contact as
table of T_Contact;
/
Create or Replace type
T_Personne as object (
prenom varchar2(30),
suit T_SET_Contact,
ecrit T_SET_Message
);
/
```

Type T_Personne

Représentation logique de la structure :

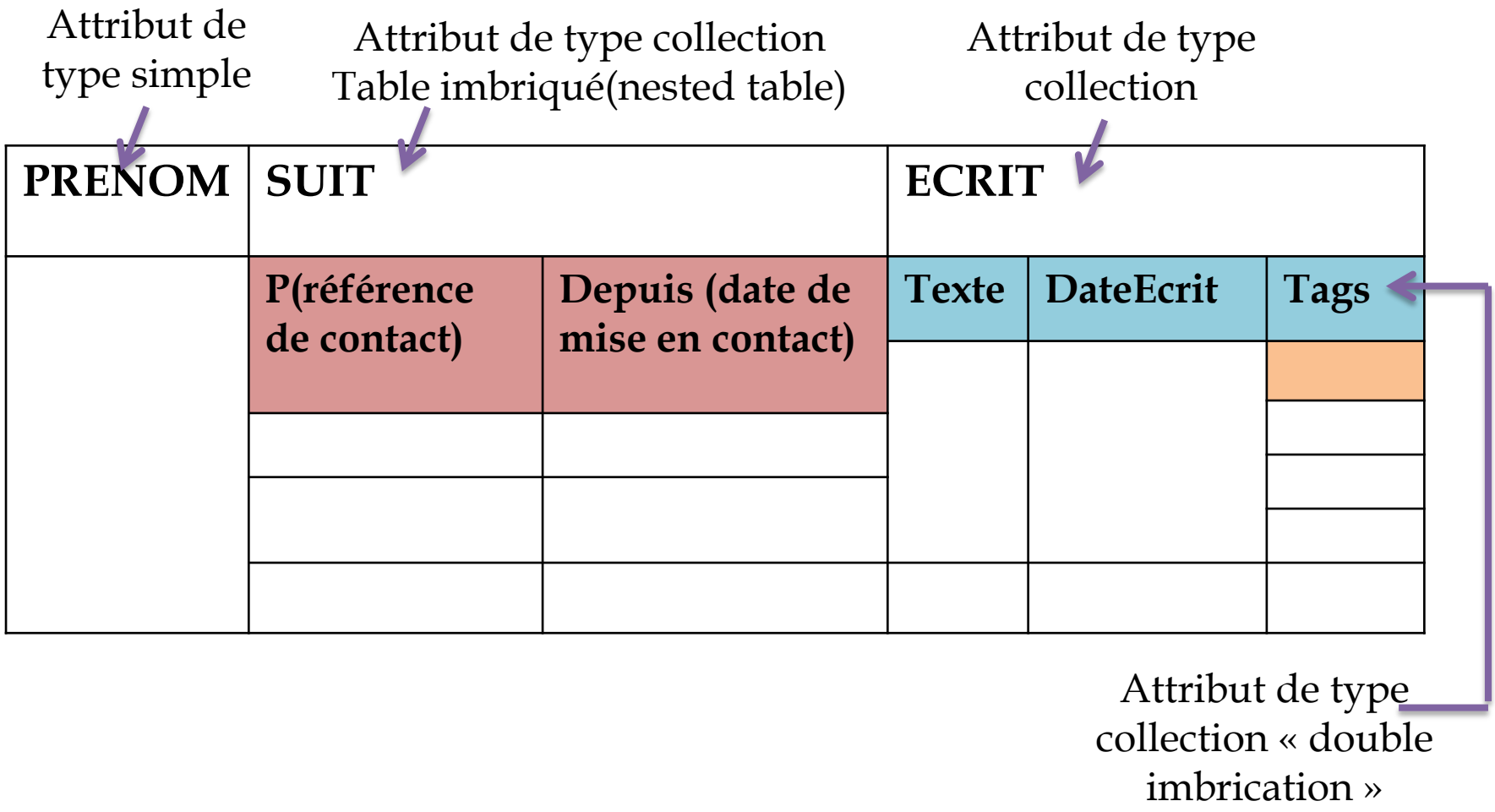


Table personne

Représentation physique:

Un tuple de la table personne

PRENOM	SUIT	ECRIT
Anis	**@1**	@2

@1

P	Depuis
Ref(Ahmed)	01/01/2020
Ref(Amine)	01/03/2020
Ref(Aya)	05/04/2020

Store(stockage) de la table imbriquée **SUIT**

@2

Texte	dateEcrit	Tags
msg1	01/12/2019	@3
msg2	01/01/2020	@4

Store de la table imbriquée **ECRIT**

tag1 @3
tag2 @4
tag3

Store de la table imbriquée **Tags**

Table personne

Script de création de la table personne:

create table personne of t_personne(primary key(prenom))

- préciser le stockage physique des attributs de type table imbriquée

-stockage de table imbriquée **suit**
nested table **suit** store as **table_suit**,

Nom physique

-stockage de la table imbriquée **ecrit** et **tags**, ici il y a une double imbrication
nested table **ecrit** store as **table_ecrit**(nested table **tags** store as **table_tags**);

Table personne

Insertion dans la table personne: (1)

La table Personne contient les personnes Amel et Badis ;
Ecrire l'instruction SQL3 permettant d'insérer dans la table Personne la personne suivante : **Manel** suit **Amel** depuis le **01/01/2017** et a écrit le message '**Oran candidat aux jeux Méditerranéens 2022**' concernant le tag '**JM2022**' le **01/06/2018**.

Table personne

Insertion dans la table personne: (2)

```
insert into personne values(t_personne('Manel',
T_SET_Contact(
t_contact(
(select ref(p) from personne p where
p.prenom='Amel'),
'11/06/2017')),
T_SET_Message(
T_Message(
'Oran candidat aux jeux Méditerranéens 2022',‘
01/06/2018',
T_SET_Tag('JM2022')
))
));
```

Table personne

Insertion dans une table imbriquée:

Ecrire l'instruction SQL3 qui insère Badis dans les contacts de Manel le 01/06/2018.

```
insert into table(select p.suit from personne p where
p.prenom='Manel')
values  (t_contact(
(select ref(p) from personne p where
p.prenom='Badis'),
'01/06/2018'));
```

Table personne

Requêtes SQL3:

a. Quels sont les prénoms des personnes que Amel suit depuis le 01/06/2018 ou une date antérieure et qui ont écrit un message contenant le tag 'JM2022' ?
Le résultat ne contient pas de doublons. La solution ne doit pas contenir de sous-requête.

Table personne

Requêtes SQL3:

```
select distinct deref(s.p).prenom
from personne pp, table(pp.suit) s,
table(deref(s.p).ecrit) m, table(m.tags) t
where pp.prenom='Amel'
And s.depuis<='01/01/2018‘
and t.column_value='JM2022';
```

Table personne

Requêtes SQL3:

b. Quelles sont les personnes qui ont écrit un message le même jour qu'Amel. Afficher des couples formés d'une date et d'un objet personne. Trier le résultat par date croissante.

Table personne

Requêtes SQL3:

```
select distinct value(p2), m2.dateEcrit
from personne p1, table(p1.ecrit) m1,
personne p2, table(p2.ecrit) m2
where p1.prenom='Amel‘ and p1.prenom< p2.prenom
and m1.dateEcrit=m2.dateEcrit
order by m1.dateEcrit asc;
```

Table personne

Requêtes SQL3:

c. Pour chaque tag, combien de personnes ont écrit au moins un message concernant ce tag ? Afficher des couples formés d'un tag et d'un nombre de personnes. Utiliser la clause group by.

Table personne

Requêtes SQL3:

```
select t.column_value, count(*) as nb

from personne p, table(p.ecrit) m, table(m.tags) t

group by t.column_value;
```

Type T_Personne

Ajout d'une méthode:

On complète le type T_Personne avec la méthode mesTags retournant l'ensemble (sans doublons) des tags des messages d'une personne. La signature de la méthode est: **member function mesTags return T_Set_Tag;**

Par exemple Badis a écrit un message ayant le tag 'Judo' et un autre message ayant les tags 'Karaté' et 'Judo'. Alors la méthode mesTags() invoquée sur Badis affiche T_SE_Tag('Karaté', 'Judo')

Type T_Personne

Ajout d'une méthode:

```
alter type t_personne
add member function mesTags return T_Set_Tag cascade;
create or replace type body t_personne
as member function mesTags return T_Set_Tag
is
tag T_Set_Tag;
Begin
Select CAST(MULTISET(
select distinct t.column_value
from personnes p, table(p.ecrit) m, table(m.tags) t
where p.prenom=self.prenom
) AS T_Set_Tag)  into tag
from dual;
 return tag;
End ;
End;
```