Bases de données Avancées : SGBD Objet-Relationnel - Partie 1

Ce TP se fait en 3 parties sur 2 séances : celle du 7/10/16 et celle du 14/10/16 (le rendu est facultatif et doit être fait avant le 14/10/16 sur Moodle). La partie 1 se focalise sur la structure des objets, la partie 2 sur le comportement des objets et la partie 3 sur l'aproche vues objets.

L'objectif de ce TP est de vous familiariser avec les bases de données objet relationnel en vue du mini-projet que vous aurez à présenter (rapport + soutenance) le 3 novembre 2016.

1 Le modèle

Nous partons du diagramme de classes UML simple ci-dessous.

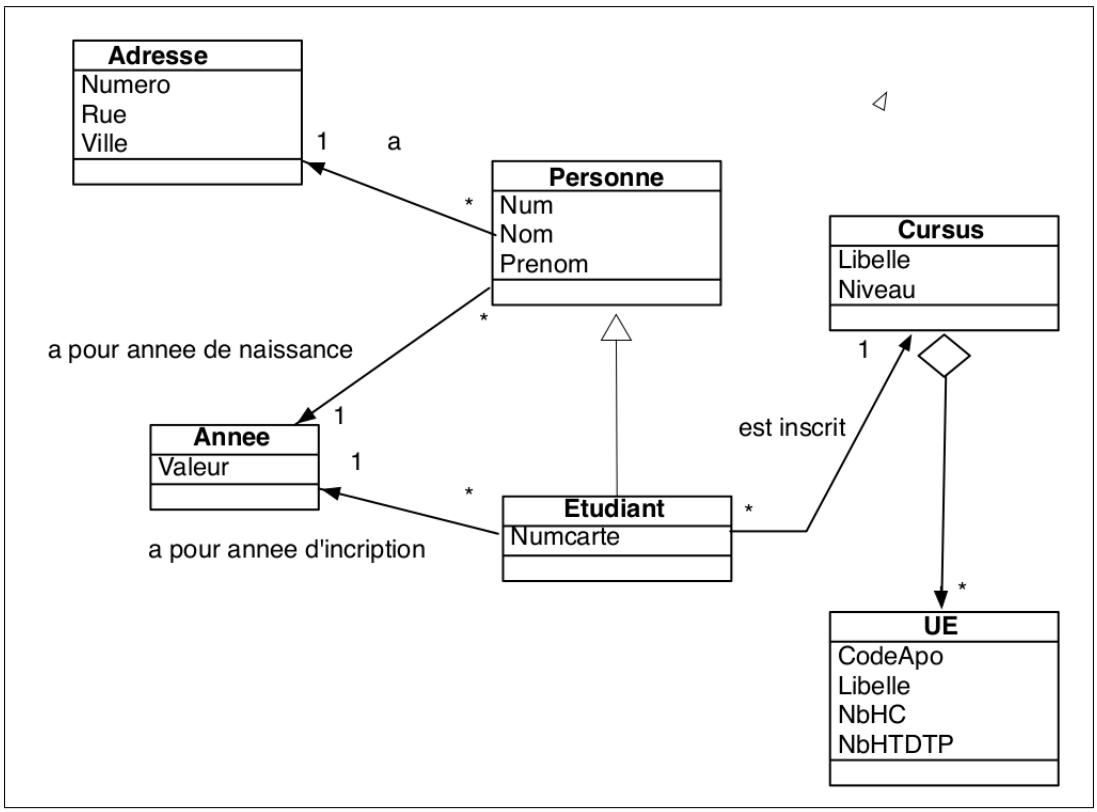

FIGURE 1 – Modèle

2 Un cas simple de modèle objet-relationnel

Pour cette partie, nous ne considérons que les 3 classes "Personne", "Adresse", et "Année" du modèle de la figure 1.

Des contraintes supplémentaires sont données :

— une personne est identifiée par son numéro NUM,
— le nom d'une personne doit être UNIQUE
— l'année de naissance d'une personne doit être comprise entre 1900 et 2007

2.1 L'approche relationnel étendu - UML / RO

L'objectif ici est de rester dans une démarche relationnel en ne considérant que les types abstraits de données les plus interessants et en gardant les tables relationnelles.

Question 1 A partir du modèle UML, déterminer les types complexes (TAD) qu'il serait intéressant de créer.

Question 2 Donner le schéma RO ("schéma relationnel étendu de type relationnel objet") résultant en décrivant les types et relations (description textuelle ou shématique).

Question 3 Sous Oracle, implémenter les types et tables correspondant au schéma défini précédemment sans oublier les contraintes spécifiées. Vérifier la description de ces types (desc) et vérifier les descriptions au sein des tables de la métabase USER_TYPES, USER_TYPE_ATTRS, USER_TABLES et USER_CONSTRAINTS.

Syntaxe : rappel de la syntaxe de création d'un type

```
create [or replace] type <nom type> AS Object
(col1 type1, col2 type2, ...)
/
```

2.2 La vision objet - UML / OR

Vous allez travailler sur le même schéma que précédemment mais, cette fois-ci, avec une vision objet. L'objectif est de tout considérer en tant que type et de définir des tables objets comme extension de ces types. Vous pouvez ré-utiliser les types définis précedemment

Question 4 A partir du modèle UML, déterminer les types complexes (TAD).

Question 5 Donner le schéma OR en décrivant les types et table objet (description textuelle ou shématique).

Question 6 Sous Oracle, implémenter les types (si différents de précédemment) et les tables objets correspondant au schéma défini précédemment sans oublier les contraintes spécifiées. Vérifier la description de ces types (desc) et vérifier les descriptions au sein des tables de la métabase USER_TYPES, USER_TYPE_ATTRS, USER_OBJECT_TABLES et USER_CONSTRAINTS.

Syntaxe : rappel de la syntaxe de création d'une table objet

```
create  table <nom table> OF <nom type> (col1 type1, .....)
```

2.3 Comparaison UML / RO - UML / OR

Pour chaque implémentation - le schéma RO et le schéma OR, effectuer les actions définies ci-dessous.

Question 7 Insérer quelques tuples / objets dans la table Personnes (habitant Montpellier, Narbonne, Béziers, Nimes), vérifier que les contraintes sont opérantes.

Question 8 Interrogations : ecrire les requêtes qui permettent d'obtenir
- la totalité des informations de toutes les personnes
- le nom et l'adresse de toutes les personnes
- les références de toutes les personnes
- le nom et l'année de naissance des personnes résidant dans la ville de Nimes

3 Les collections dans le modèle objet-relationnel

Pour cette partie, nous ne considérons que les 2 classes "Cursus" et "UE" du modèle de la figure 1. Dans ce modèle, un cursus est composé de plusieurs UE.

Question 1 Pour chaque approche (RO et OR), déterminer les types complexes (TAD) et donner les schémas RO/OR

Indication : dans l'approche RO, le cursus est vue comme une relation dont un des attributs est la collection d'UE (cette collection est un type pré-défini), alors que dans l'approche OR, le cursus est vue comme un objet de type Cursus (le type Cursus ayant la collection d'UE comme attribut).

Question 2 Sous Oracle, implémenter le schéma de l'approche OR (vision objet). Pour les collections d'UE, vous implémenterez les 2 solutions (tableau prédimensionné et collections illimitées) afin de les comparer.

Syntaxe : deux types de collections sont possibles dans Oracle : les tableaux prédimensionnés (varray) et les collections illimitées qui seront utilisées comme tables imbriquées (nested tables).

```
create [or replace] type <nom type collection> AS VARRAY(dim) OF  <nom type>
/

create [or replace] type <nom type collection> AS Table OF  <nom type>
/
```

Question 3 Insertions (à réaliser pour les 2 solutions de collection)

- Insérer des cursus et leur collection d'UE . Par exemple,

```
AIGLE, M1
et la collection d'UE
HMIN106M BDA  15 36
HMIN116 Reseau 10 40
HMIN112M SIBD 12 38

AIGLE, M2
et la collection d'UE
HMIN305 Complexite 20 0
HMIN314 Web 10 10
HMIN303 Genie logiciel 20 0

etc...
```

- Effectuer des insertions d'UE complémentaires dans des cursus existants (par exemple ajouter HMIN105 IA 20 20 dans le cursus AIGLE M1).

Question 4 Interrogations (à réaliser pour les 2 solutions de collection) : ecrire les requêtes qui permettent d'obtenir

- le libellé, le niveau et la collection d'Ue de chaque cursus
- les codes Apogée des UE du cursus de M1 de libellé 'AIGLE'
- les libellés des cursus et les codes Apogée des UE ayant un nombre d'heures de cours > 15 heures

4 La spécialisation et les références dans le modèle objet-relationnel

Pour cette partie, nous allons considérer l'ensemble du modèle de la figure 1. Nous allons donc traiter le cas de la classe "Etudiant" qui spécialise "Personne" et qui référence un "Cursus". Nous utiliserons l'implémentation faite dans la 3eme partie ainsi que l'implémentation de l'approche OR de la 2eme partie.

Question 1 En utilisant une approche OR, déterminer les types complexes (TAD) et donner le schéma OR.

Question 2 Sous Oracle, implémenter ce shéma.

Indication : pour permettre la spécialisation, la clause NOT FINAL doit être ajoutée au type générique ; un type ne peut être modifié si des tables ou d'autres types l'utilisent.

Syntaxe : création d'un type qui spécialise un autre type

```
Create type <nomtype> UNDER <nomtypegeneral>
(col1 type 1, ...)
Not final /Final
/
```

Question 3 Réaliser des insertions dans cette table. On peut commencer à insérer des valeurs correspondant à des valeurs de la table Personnes et initialiser le cursus à 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 prélablement créés (via la référence objet correspondante).

Question 4 Interrogations : écrire les requêtes qui permettent d'obtenir
- le nom, l'annee d'inscription et le cursus des étudiants
- le nom et le code Apogée des UE suivies par un étudiant donné

5 Poue aller plus loin

Nous pourrions maintenant envisager d'établir la relation bidirectionnelle entre Etudiant et Cursus : un etudiant référence son cursus, un cursus référence tous les étudiants qui le suivent.

Question 1 Comment pouvez-vous procéder ?