Département Informatique et Statistique, Faculté de SEG, Université Lumière Lyon 2
M2 Informatique spécialité IDS-IIDEE – Année 2011-2012
Bases de données objets – Examen
J. Darmont (*http://eric.univ-lyon2.fr/~jdarmont/*), 04/04/12

Durée : 2 heures – Documents autorisés – Barème fourni à titre indicatif

Exercice 1 (3 points)

On dispose de données stockées sous forme relationnelle, utilisées par des applications existantes, que l'on souhaite exploiter dans un nouveau programme purement orienté objets (Java, C#…) sans passer par une couche intergicielle (*middleware*). Quelle stratégie est la plus adaptée pour que ce programme puisse accéder à ces données (justifier en quelques mots) :

1. conserver les données dans des tables relationnelles ;
2. migrer les données dans des tables objets ;
3. construire des vues objets sur les tables relationnelles ?

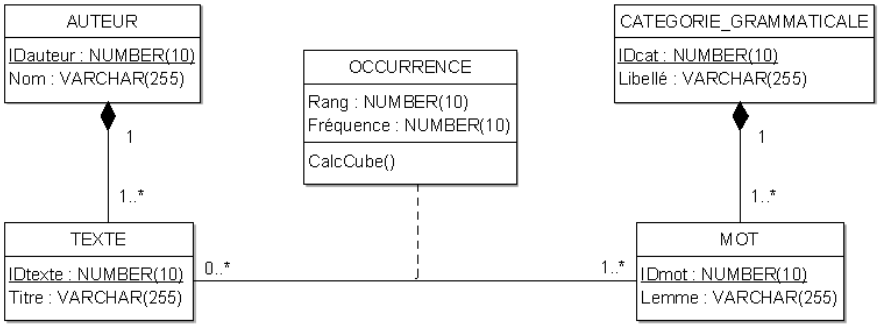
Exercice 2 (5 points)

Soit la classe C1 préexistante, dont les attributs sont A1 et A2. Ajouter à C1 une fonction ORDER permettant de calculer une similarité rudimentaire* entre deux instances X et Y de C1. Donner un exemple d'appel à cette fonction depuis un programme PL/SQL.

*
```
Si X.A1 = Y.A1 alors
     Si X.A2 = Y.A2 alors
          sim = 1
     Sinon
          sim = 0.5
     Fsi
Sinon
     Si X.A2 = Y.A2 alors
          sim = 0.5
     Sinon
          sim = 0
     Fsi
Fsi
```

Exercice 3 (12 points)

Soit le magasin de données linguistiques dont le modèle conceptuel (diagramme de classes UML) est fourni ci-contre et qui permet d'observer des occurrences de mots dans des textes. Dans ce modèle, le rang d'un mot dans un texte représente une dimension temporelle dégénérée.



La méthode CalcCube de la classe-association OCCURRENCE, qui représente les faits de ce magasin, permet de calculer sous forme de cube OLAP la moyenne des rangs et la somme des fréquences de chaque mot (lemme) en fonction des auteurs (noms), et de les afficher au format `(Mot, Auteur) : Rang_Moyen, Fréquence_Totale`.

Proposer un modèle physique Oracle relationnel-objet du magasin de données linguistiques, sous forme de code PL/SQL et de requêtes SQL, incluant l'implémentation de la méthode CalcCube.

Correction exercice 1

La solution 1 n'est pas bonne car les données relationnelles de la base doivent cohabiter avec les objets de l'application.

La solution 2 implique soit une maintenance coûteuse de la base objets dupliquée, soit une perte de compatibilité avec les applications existantes qui utilisent les données sous forme relationnelle.

La solution 3, à retenir, permet d'interfacer les données relationnelles avec l'application orientée objets.

Correction exercice 2

```
-- Ajout de la function ORDER dans la spécification du type
ALTER TYPE C1 ADD ORDER MEMBER FUNCTION Sim(c C1) RETURN NUMBER;

-- Ajout de la function ORDER dans le corps du type
CREATE OR REPLACE TYPE BODY C1 AS
    -- Redéfinition des methodes existantes si nécessaire
    ORDER MEMBER FUNCTION Sim(c C1) RETURN NUMBER IS
      BEGIN
         IF SELF.A1 = c.A1 THEN
            IF SELF.A2 = c.A2 THEN
               RETURN 1;
            ELSE
               RETURN 0.5;
            END IF;
         ELSE
            IF SELF.A2 = c.A2 THEN
               RETURN 0.5;
            ELSE
               RETURN 0;
            END IF;
         END IF;
      END;
END;
/

-- Bloc anonyme de test
DECLARE
     X C1;
     Y C1;
BEGIN
     X := NEW C1(10, 20);
     Y := NEW C1(10, 21);
     DBMS_OUTPUT.PUT_LINE('Similarité(X, Y) = ' || X.Sim(Y));
END;
/
```

Correction exercice 3

```
-- Types

CREATE TYPE T_AUTEUR AS OBJECT(
  IDauteur NUMBER(10),
  Nom VARCHAR(255))
/
```



```
CREATE TYPE T_TEXTE AS OBJECT(
  IDtexte NUMBER(10),
  Titre VARCHAR(255),
  REFauteur REF T_AUTEUR)
/

CREATE TYPE T_CATEGORIE AS OBJECT(
  IDcat NUMBER(10),
  Libelle VARCHAR(255))
/

CREATE TYPE T_MOT AS OBJECT(
  IDmot NUMBER(10),
  Lemme VARCHAR(255),
  REFcat REF T_CATEGORIE)
/

CREATE TYPE T_OCCURRENCE AS OBJECT(
  REFtexte REF T_TEXTE,
  REFmot REF T_MOT,
  Rang NUMBER(10),
  Frequence NUMBER(10),
  STATIC PROCEDURE CalcCube)
/


-- Tables

CREATE TABLE O_AUTEUR OF T_AUTEUR(
  CONSTRAINT o_auteur_pk PRIMARY KEY(IDauteur));

CREATE TABLE O_TEXTE OF T_TEXTE(
  CONSTRAINT o_texte_pk PRIMARY KEY(IDtexte),
  CONSTRAINT o_texte_refauth REFauteur REFERENCES O_AUTEUR);

CREATE TABLE O_CATEGORIE OF T_CATEGORIE(
  CONSTRAINT o_categorie_pk PRIMARY KEY(IDcat));

CREATE TABLE O_MOT OF T_MOT(
  CONSTRAINT o_mot_pk PRIMARY KEY(IDmot),
  CONSTRAINT o_mot_refcat REFcat REFERENCES O_CATEGORIE);

CREATE TABLE O_OCCURRENCE OF T_OCCURRENCE(
  CONSTRAINT o_occurrence_reftexte REFtexte REFERENCES O_TEXTE,
  CONSTRAINT o_occurrence_refmot REFmot REFERENCES O_MOT);

CREATE OR REPLACE TRIGGER o_occurrence_pk
BEFORE INSERT OR UPDATE ON O_OCCURRENCE
FOR EACH ROW
  DECLARE
    n INTEGER;
    e_null EXCEPTION;
    e_exists EXCEPTION;
  BEGIN
    IF :NEW.REFtexte IS NULL OR :NEW.REFmot IS NULL THEN
      RAISE e_null;
    END IF;
    SELECT COUNT(*) INTO n
      FROM O_OCCURRENCE
      WHERE REFtexte = :NEW.REFtexte AND REFmot = :NEW.REFmot;
    IF n > 0 THEN
      RAISE e_exists;
    END IF;
```

```
  EXCEPTION
    WHEN e_null THEN
      RAISE_APPLICATION_ERROR(-20501, 'Aucune partie de la clé ne doit être
NULLe !');
    WHEN e_exists THEN
      RAISE_APPLICATION_ERROR(-20502, 'La clé existe déjà dans la table !');
  END;
/

-- Corps du type T_OCCURRENCE

CREATE OR REPLACE TYPE BODY T_OCCURRENCE AS
  STATIC PROCEDURE CalcCube IS
    CURSOR c IS
      SELECT DECODE(GROUPING(o.REFmot.Lemme), 1, 'ALL', o.REFmot.Lemme) Mot,
             DECODE(GROUPING(o.REFtexte.REFauteur.Nom), 1, 'ALL',
                                                 o.REFtexte.REFauteur.Nom) Auteur,
             AVG(Rang) Rang_Moyen,
            SUM(Frequence) Frequence_Totale
      FROM O_OCCURRENCE o
      GROUP BY CUBE(o.REFmot.Lemme, o.REFtexte.REFauteur.Nom);
    BEGIN
      FOR t IN c LOOP
        DBMS_OUTPUT.PUT_LINE('(' || t.Mot || ', ' || t.Auteur || ') : ' ||
                             t.Rang_Moyen || ', ' || t.Frequence_Totale);
      END LOOP;
    END;
END;
/
```