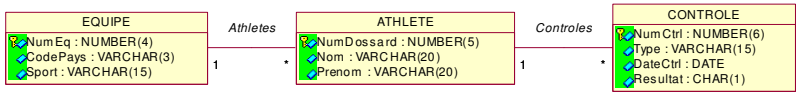
Département Informatique et Statistique, Faculté de SEG, Université Lumière Lyon 2
M2 Informatique spécialité IDS-IIDEE – Année 2011-2012

Bases de données objets – TD n° 3 : Collections

J. Darmont (*http://eric.univ-lyon2.fr/~jdarmont/*), 13/03/12

Exercice 1 : Tables imbriquées

On souhaite implanter sous Oracle une base de données permettant de faire le suivi de sportifs lors de compétitions internationales multisports (type jeux olympiques) du point de vue des contrôles antidopage. L'analyse conceptuelle des besoins exprimés par les instances chargées de ces contrôles est donnée ci-dessous sous forme d'un diagramme de classes UML.



Base de données

1. Créer un ensemble de types correspondant au diagramme de classes ci-dessus. Les associations 1–N du modèle doivent être traduites au niveau physique sous forme de collections multiniveaux. Convention de nommage des types : préfixer le nom des classes par T_ (ex. T_Athlete) et des collections par TAB_ (ex. TAB_Controle).

2. Définir une table nommée Equipe d'objets de type T_Equipe. Afficher la structure de cette table ainsi que celle de ses tables imbriquées.

3. Peupler la table Equipe avec les données ci-dessous, puis afficher son contenu.

NumEq	CodePays	Sport	NumDossard	Nom	Prénom	NumCtrl	Type	DateCtrl	Result.
1	USA	100 mètres	1	Green	Maurice	1	Sanguin	03/11/05	N
			2	Lewis	Carl	2	Sanguin	03/11/05	N
2	USA	Basket	23	Jordan	Michael	3	Urinaire	03/11/05	N
			8	Bryant	Kobe	4	Urinaire	03/11/05	N
			34	O Neal	Shaquille	5	Urinaire	03/11/05	N
3	UK	100 mètres	4	Sphinx	Le	6	Sanguin	03/11/05	P
						7	Sanguin	04/11/05	N
4	UK	Aviron	5	Smith	John	8	Urinaire	03/11/05	N
			6	Smith	Jack	9	Urinaire	03/11/05	P
						10	Urinaire	04/11/05	P
5	FRA	Aviron	9	Dupond	Albert	11	Urinaire	03/11/05	N
			7	Martin	Maurice	12	Urinaire	03/11/05	N
6	FRA	Basket	10	Bilba	Jim	13	Urinaire	03/11/05	N
			11	Parker	Tony	14	Urinaire	03/11/05	N
			12	Diaw	Boris				

Requêtes de « désemboîtement »

1. Liste des équipes (CodePays, Sport) avec les athlètes (Nom, Prénom) les constituant (effectuer une pseudojointure).

2. Idem en utilisant un curseur imbriqué. Conclusion ?

3. Nom et prénom des athlètes triés par pays et par ordre alphabétique.

4. Numéro et date des contrôles antidopage effectués par chaque équipe (pseudojointures).

5. Idem en utilisant des curseurs imbriqués.

6. Nom et prénom des athlètes de l'équipe n° 2.

7. Liste des contrôles effectués par l'athlète n° 4 de l'équipe n° 3.

8. Nombre d'athlètes dans l'équipe n° 6.

9. Nombre total d'athlètes par pays.

10. Nombre de contrôles par athlète (ordre alphabétique).

11. Nombre de contrôles positifs par athlète (ordre alphabétique).

12. Nombre de contrôles positifs par pays (ordre alphabétique).

13. Date de dernier contrôle pour chaque athlète (ordre alphabétique).

14. Équipe(s) les plus contrôlées.

15. Équipes dont aucun athlète n'a été contrôlé positif.

Mises à jour des tables imbriquées

1. Insérer l'athlète <30, 'Pietrus', 'Michael', *pas de contrôle*> dans l'équipe n° 6.

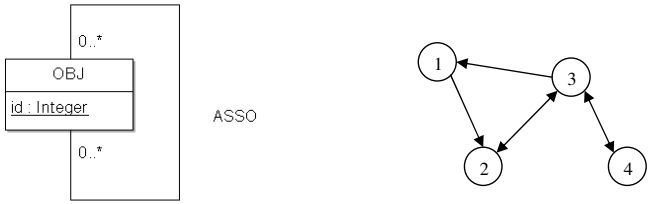
2. Insérer les contrôles <15, 'Urinaire', '23/01/2007', 'N'> et <16, 'Sanguin', '23/01/2007', 'N'> pour l'athlète n° 30 de l'équipe n° 6.

3. Supprimer tous les contrôles de l'athlète n° 4 de l'équipe n° 3.

4. Modifier à « Négatif » le résultat des contrôles de l'athlète n° 6 de l'équipe n° 4 passés avant le 04/11/2005.

Exercice 2 : Classes, associations et intégrité référentielle

Soient la classe OBJ et l'association récursive ASSO représentées dans la partie gauche de la figure ci-contre, qui permettent de représenter la structure des liens orientés entre les objets de la partie droite de la figure. L'objectif de cet exercice est de comparer deux représentations logiques/physiques de ce modèle conceptuel.



Solution 1 : Type et table-association

1.1. Créer un type OBJ pour représenter les objets et un type ASSO constitué de deux références vers des objets OBJ pour représenter l'association récursive.

1.2. Créer une table OBJ1 d'objets de type OBJ et une table ASSOBJ d'objets de type ASSO. Ne pas oublier les contraintes d'intégrité (clé primaire d'OBJ1, contraintes de référence et de non nullité des références d'ASSOBJ).

1.3. Peupler les tables OBJ1 et ASSOBJ. Insérer deux objets dans ASSOBJ pour représenter les relations bidirectionnelles entre objets. Par exemple, la relation entre l'objet 3 et l'objet 4 donne lieu à deux occurrences : (3, 4) et (4, 3) (lire « de 3 vers 4 et de 4 vers 3 », respectivement).

1.4. Insérer une deuxième fois la relation (3, 4) dans ASSOBJ. Conclusion ?

1.5. Insérer la relation (3, 9) dans ASSOBJ. Conclusion ?

1.6. Insérer la relation (3, NULL) dans ASSOBJ. Conclusion ?

1.7. Supprimer l'objet 1 dans OBJ1. Conclusion ?

Solution 2 : Collection de références

2.1. Créer un type OBJ_V2 vide et un type TAB_OBJ table de références vers des objets OBJ_V2. Compléter le type OBJ_V2 avec les attributs ID de type NUMBER(2) et ASSO_V2 de type TAB_OBJ.

2.2. Créer une table OBJ2 d'objets de type OBJ_V2 ainsi qu'une table imbriquée ASSOBJ2 pour stocker la collection ASSO_V2.

2.3. Peupler la table OBJ2 avec les identifiants d'objets et la collection ASSO_V2 vide. Une fois les objets créés (et donc susceptibles d'être référencés), peupler les collections ASSO_V2 de chaque objet en prenant en compte le caractère bidirectionnel ou non des relations entre objets.

2.4. Insérer une deuxième fois une référence à l'objet 4 dans la collection dans ASSO_V2 de l'objet 3 d'OBJ2. Conclusion ?

2.5. Insérer une référence à l'objet 9 dans la collection dans ASSO_V2 de l'objet 3 d'OBJ2. Conclusion ?

2.6. Insérer une référence NULLe dans la collection dans ASSO_V2 de l'objet 3 d'OBJ2. Conclusion ?

2.7. Supprimer l'objet 1 dans OBJ2. Conclusion ?

Conclusion générale

3.1. Sur la facilité de gestion de l'association récursive avec ces deux solutions ?

3.2. Sur la gestion de l'intégrité référentielle avec ces deux solutions ?

Correction Exercice 1

-- Types

```
CREATE TYPE T_Controle AS OBJECT(
     NumCtrl NUMBER(6),
     Type VARCHAR(15),
     DateCtrl DATE,
     Resultat CHAR(1))
/

CREATE TYPE TAB_Controle AS TABLE OF T_Controle
/

CREATE TYPE T_Athlete AS OBJECT(
     NumDossard NUMBER(5),
     Nom VARCHAR(20),
     Prenom VARCHAR(20),
     Controles TAB_Controle)
/

CREATE TYPE TAB_Athlete AS TABLE OF T_Athlete
/

CREATE TYPE T_Equipe AS OBJECT(
     NumEq NUMBER(4),
     CodePays VARCHAR(3),
     Sport VARCHAR(15),
     Athletes TAB_Athlete)
/
```

-- Table

```
CREATE TABLE Equipe OF T_Equipe (CONSTRAINT pk_equipe PRIMARY KEY(NumEq))
NESTED TABLE Athletes STORE AS Stockage_athletes
     (NESTED TABLE Controles STORE AS Stockage_controles);

DESC Equipe
DESC Stockage_athletes
DESC Stockage_controles

INSERT INTO Equipe VALUES(1, 'USA', '100 metres',
     TAB_Athlete(T_Athlete(1, 'Green', 'Maurice',
          TAB_Controle(T_Controle(1, 'Sanguin', '3/11/2005', 'N'))),
     T_Athlete(2, 'Lewis', 'Karl',
          TAB_Controle(T_Controle(2, 'Sanguin', '3/11/2005', 'N')))));

INSERT INTO Equipe VALUES(2, 'USA', 'Basket',
     TAB_Athlete(T_Athlete(23, 'Jordan', 'Michael',
          TAB_Controle(T_Controle(3, 'Urinaire', '3/11/2005', 'N'))),
     T_Athlete(8, 'Bryant', 'Kobe',
          TAB_Controle(T_Controle(4, 'Urinaire', '3/11/2005', 'N'))),
     T_Athlete(34, 'O Neal', 'Shaquille',
          TAB_Controle(T_Controle(5, 'Urinaire', '3/11/2005', 'N')))));

INSERT INTO Equipe VALUES(3, 'UK', '100 metres',
     TAB_Athlete(T_Athlete(4, 'Sphynx', 'Le',
          TAB_Controle(T_Controle(6, 'Sanguin', '3/11/2005', 'P'),
          T_Controle(7, 'Sanguin', '4/11/2005', 'N')))));
```



```
INSERT INTO Equipe VALUES(4, 'UK', 'Aviron',
     TAB_Athlete(T_Athlete(5, 'Smith', 'John',
          TAB_Controle(T_Controle(8, 'Urinaire', '3/11/2005', 'N'))),
     T_Athlete(6, 'Smith', 'Jack',
          TAB_Controle(T_Controle(9, 'Urinaire', '3/11/2005', 'P'),
          T_Controle(10, 'Urinaire', '4/11/2005', 'P')))));

INSERT INTO Equipe VALUES(5, 'FRA', 'Aviron',
     TAB_Athlete(T_Athlete(9, 'Dupond', 'Albert',
          TAB_Controle(T_Controle(11, 'Urinaire', '3/11/2005', 'N'))),
     T_Athlete(7, 'Martin', 'Maurice',
          TAB_Controle(T_Controle(12, 'Urinaire', '3/11/2005', 'N')))));

INSERT INTO Equipe VALUES(6, 'FRA', 'Basket',
     TAB_Athlete(T_Athlete(10, 'Bilba', 'Jim',
          TAB_Controle(T_Controle(13, 'Urinaire', '3/11/2005', 'N'))),
     T_Athlete(11, 'Parker', 'Tony',
          TAB_Controle(T_Controle(14, 'Urinaire', '3/11/2005', 'N'))),
          T_Athlete(12, 'Diaw', 'Boris', TAB_Controle())));

SELECT * FROM Equipe;
```

-- Requêtes

```
-- 1
SELECT e.CodePays, e.Sport, a.Nom, a.Prenom
FROM Equipe e, TABLE(e.Athletes) a;

-- 2
SELECT e.CodePays, e.Sport,
     CURSOR(SELECT a.Nom, a.Prenom FROM TABLE(e.Athletes) a)
FROM Equipe e;

-- 3
SELECT e.CodePays, a.Nom, a.Prenom
FROM Equipe e, TABLE(e.Athletes) a
ORDER BY e.CodePays, a.Nom, a.Prenom;

-- 4
SELECT e.NumEq, c.NumCtrl, c.DateCtrl
FROM Equipe e, TABLE(e.Athletes) a, TABLE(a.Controles) c;

-- 5
SELECT e.NumEq,
     CURSOR(SELECT
          CURSOR(SELECT c.NumCtrl, c.DateCtrl FROM TABLE(a.Controles) c)
          FROM TABLE(e.Athletes) a)
FROM Equipe e;

-- 6
SELECT a.Nom, a.Prenom
FROM TABLE (SELECT Athletes FROM Equipe WHERE NumEq = 2) a;

-- 7
SELECT *
FROM TABLE (SELECT Controles
     FROM TABLE (SELECT Athletes FROM Equipe WHERE NumEq = 3)
     WHERE NumDossard = 4);

-- 8
SELECT COUNT(*)
FROM TABLE (SELECT Athletes FROM Equipe WHERE NumEq = 6);
```

```
-- 9
SELECT CodePays, COUNT(*)
FROM Equipe e, TABLE (e.Athletes)
GROUP BY CodePays;

-- 10
SELECT a.Nom, a.Prenom, COUNT(*)
FROM Equipe e, TABLE(e.Athletes) a, TABLE(a.Controles)
GROUP BY a.Nom, a.Prenom
ORDER BY a.Nom, a.Prenom;

-- 11
SELECT a.Nom, a.Prenom, COUNT(*)
FROM Equipe e, TABLE(e.Athletes) a, TABLE(a.Controles) c
WHERE c.Resultat = 'P'
GROUP BY a.Nom, a.Prenom
ORDER BY a.Nom, a.Prenom;

-- 12
SELECT e.CodePays, COUNT(*)
FROM Equipe e, TABLE(e.Athletes) a, TABLE(a.Controles) c
WHERE c.Resultat = 'P'
GROUP BY e.CodePays
ORDER BY e.CodePays;

-- 13
SELECT a.Nom, a.Prenom, MAX(c.DateCtrl)
FROM Equipe e, TABLE(e.Athletes) a, TABLE(a.Controles) c
GROUP BY a.Nom, a.Prenom
ORDER BY a.Nom, a.Prenom;

-- 14
SELECT e.NumEq, e.CodePays, e.Sport
FROM Equipe e, TABLE(e.Athletes) a, TABLE(a.Controles)
GROUP BY e.NumEq, e.CodePays, e.Sport
HAVING COUNT(*) =
     (SELECT MAX(COUNT(*))
     FROM Equipe e1, TABLE(e1.Athletes) a1, TABLE(a1.Controles)
     GROUP BY e1.NumEq);

-- 15
SELECT NumEq, CodePays, Sport
FROM Equipe e1
WHERE NOT EXISTS(
     SELECT *
     FROM Equipe e2, TABLE(e2.Athletes) a, TABLE(a.Controles) c
     WHERE e1.NumEq = e2.NumEq
     AND c.Resultat = 'P');
```

-- Mises à jour

```
-- 1
INSERT INTO TABLE (SELECT Athletes FROM Equipe WHERE NumEq = 6)
VALUES(30, 'Pietrus', 'Michael', TAB_Controle());

-- 2
INSERT INTO TABLE ( SELECT a.Controles FROM Equipe, TABLE(Athletes) a
                    WHERE NumEq = 6 AND NumDossard = 30)
VALUES(15, 'Urinaire', '23/01/2007', 'N');

INSERT INTO TABLE ( SELECT a.Controles FROM Equipe, TABLE(Athletes) a
                    WHERE NumEq = 6 AND NumDossard = 30)
VALUES(16, 'Sanguin', '23/01/2007', 'N');
```



```
-- 3
DELETE FROM TABLE ( SELECT a.Controles FROM Equipe, TABLE(Athletes) a
                    WHERE NumEq = 3 AND NumDossard = 4);

-- 4
UPDATE TABLE (SELECT a.Controles FROM Equipe, TABLE(Athletes) a
              WHERE NumEq = 4 AND NumDossard = 6)
SET Resultat = 'N'
WHERE DateCtrl < '04/11/2005';
```

Correction Exercice 2

```
-- Solution 1 : classes

create or replace type obj as object(
  id number(2)
)
/

create or replace type asso as object(
  ref1 ref obj,
  ref2 ref obj
)
/

-- Solution 1 : tables objets

create table obj1 of obj(constraint o1pk primary key(id));

create table assobj of asso(
     constraint cref1 ref1 references obj1,
     constraint nn1 check(ref1 is not null),
     constraint cref2 ref2 references obj1,
     constraint nn2 check(ref2 is not null));

-- Solution 1 : alimentation de la base

insert into obj1 values(1);
insert into obj1 values(2);
insert into obj1 values(3);
insert into obj1 values(4);

insert into assobj values(
  (select ref(o) from obj1 o where o.id = 1),
  (select ref(o) from obj1 o where o.id = 2)
);
insert into assobj values(
  (select ref(o) from obj1 o where o.id = 2),
  (select ref(o) from obj1 o where o.id = 3)
);
insert into assobj values(
  (select ref(o) from obj1 o where o.id = 3),
  (select ref(o) from obj1 o where o.id = 1)
);
insert into assobj values(
  (select ref(o) from obj1 o where o.id = 3),
  (select ref(o) from obj1 o where o.id = 2)
);
insert into assobj values(
  (select ref(o) from obj1 o where o.id = 3),
  (select ref(o) from obj1 o where o.id = 4)
);
```

```
insert into assobj values(
  (select ref(o) from obj1 o where o.id = 4),
  (select ref(o) from obj1 o where o.id = 3)
);

-- Solution 1 : tests d'intégrité

-- Unicité de l'identifiant de l'association
-- Ne doit pas fonctionner : problème ! Un trigger est nécessaire.
insert into assobj values(
  (select ref(o) from obj1 o where o.id = 3),
  (select ref(o) from obj1 o where o.id = 4)
);

-- Référence inexistante
-- Ne doit pas fonctionner : OK
insert into assobj values(
  (select ref(o) from obj1 o where o.id = 3),
  (select ref(o) from obj1 o where o.id = 9)
);

-- Référence NULLe
-- Ne doit pas fonctionner : OK
insert into assobj values(
  (select ref(o) from obj1 o where o.id = 3),
  NULL
);

-- Destruction d'objet référencé
-- Ne doit pas fonctionner : OK
delete from obj1 where id = 1;

-- Solution 2 : classes

create or replace type obj_v2
/

create or replace type tab_obj as table of ref obj_v2
/

create or replace type obj_v2 as object(
  id number(2),
  asso_v2 tab_obj
)
/

-- Solution 2 : table objet

create table obj2 of obj_v2(constraint o2pk primary key(id)) nested table
asso_v2 store as assobj2;

-- Solution 2 : alimentation de la base

insert into obj2 values(1, tab_obj());
insert into obj2 values(2, tab_obj());
insert into obj2 values(3, tab_obj());
insert into obj2 values(4, tab_obj());

insert into table (select o.asso_v2 from obj2 o where o.id = 1) values(
  (select ref(o) from obj2 o where o.id = 2));
insert into table (select o.asso_v2 from obj2 o where o.id = 2) values(
  (select ref(o) from obj2 o where o.id = 3));
insert into table (select o.asso_v2 from obj2 o where o.id = 3) values(
  (select ref(o) from obj2 o where o.id = 1));
```



```
insert into table (select o.asso_v2 from obj2 o where o.id = 3) values(
  (select ref(o) from obj2 o where o.id = 2));
insert into table (select o.asso_v2 from obj2 o where o.id = 3) values(
  (select ref(o) from obj2 o where o.id = 3));

insert into table (select o.asso_v2 from obj2 o where o.id = 4) values(
  (select ref(o) from obj2 o where o.id = 3));

-- Solution 2 : tests d'intégrité

-- Unicité de l'identifiant
-- Ne doit pas fonctionner : problème ! Un trigger est nécessaire.
insert into table (select o.asso_v2 from obj2 o where o.id = 3) values(
  (select ref(o) from obj2 o where o.id = 4));

-- Référence inexistante
-- Ne doit pas fonctionner : problème ! Un trigger est nécessaire.
insert into table (select o.asso_v2 from obj2 o where o.id = 3) values(
  (select ref(o) from obj2 o where o.id = 9));

-- Référence NULLe
-- Ne doit pas fonctionner : problème ! Un trigger est nécessaire.
insert into table (select o.asso_v2 from obj2 o where o.id = 3) values(NULL);

-- Destruction d'objet référencé
-- Ne doit pas fonctionner : problème ! Un trigger est nécessaire.
delete from obj2 where id = 1;
```