Lif10 – Fondements des bases de données
TP2 – Contraintes et dictionnaire

Licence informatique – Automne 2013–2014

`http://liris.cnrs.fr/~mplantev/doku/doku.php?id=lif10`

Résumé

Ce TP poursuit le précédent avec les contraintes de clef, de valeurs non-nulles, sur les tuples et d'intégrité référentielle, qui sont les contreparties pratiques des dépendances fonctionnelles et d'inclusion vues en cours et en TD. Pour finir, une prise en main de la syntaxe PL/SQL.

Un compte rendu (requêtes sql et commentaires) sous la forme d'un fichier .txt est à déposer sur spiral à l'issue de la séance.

Exercice 1 : poursuite du TP1

Reprendre le TP précédent afin de s'assurer que vous être capable :
- de répondre convenablement à la question 3 de l'exercice 3,
- de répondre convenablement à la question 6 de l'exercice 4,
- d'effectuer l'exercice 6 en intégralité.

1 Contraintes de clé

Exercice 2 : re-création de la base avec contraintes

Supprimer les tables créées dans le TP1. Modifier ensuite les déclarations de création de tables afin de prendre en compte les contraintes suivantes :

I : Partie Contraintes de clés (`PRIMARY KEY, UNIQUE`)

1. *mID* est une clé de *Movie*.
2. Le couple (*title*, *year*) est également une clé.
3. *rID* est une clé de *Reviewer*.
4. (*rID*, *mID*, *ratingDate*) est une clé de *Rating*, mais avec des valeurs nulles autorisées.
5. Traduire les contraintes des questions précédentes dans le formalisme des dépendances fonctionnelles.

II : Partie Contraintes de valeurs non-nulles (`NOT NULL`)

1. *Reviewer.name* doit être non nul.
2. *Rating.stars* doit être non nul.

III : Partie Contraintes sur les tuples (`CHECK`)

1. Les films doivent être réalisés après 1900.
2. *Rating.stars* est défini sur l'ensemble $\{1, 2, 3, 4, 5\}$.
3. *Rating.ratingDate* doit prendre une valeur supérieure ou égale à 2000.
4. (†) Les films réalisés par *Steven Spielberg* sont antérieurs à 1990 et ceux de *James Cameron* sont postérieurs à 1990

IV : Partie Modification de schéma sans suppression (`ALTER TABLE`)

1. reprendre les questions précédentes en modifiant la structure des tables sans les supprimer.

Exercice 3 : vérification des contraintes

I : PartieVérifier que chacune des commandes suivante génère une erreur. Préciser la contrainte violée en relevant le message d'erreur obtenu.

1. `update Movie set mID = mID +1;`
2. `insert into Movie values (109, 'Titanic', 1997, 'JC');`
3. `insert into Reviewer values (201, 'Ted Codd');`
4. `update Rating set rID = 205, mID=104;`
5. `insert into Reviewer values (209, null);`
6. `update Rating set stars = null where rID = 208;`
7. `update Movie set year = year - 40;`
8. `update Rating set stars = stars + 1;`
9. `insert into Rating values (201, 101, 1, '1999-01-01');`
10. `insert into Movie values (109, 'Jurassic Park', 1993, 'Steven Spielberg');`
11. `update Movie set year = year-10 where title = 'Titanic';`

II : PartieVérifier que chacune des commandes suivantes ne génère pas d'erreur.

1. `insert into Movie values (109, 'Titanic', 2001, null);`
2. `update Rating set mID = 109;`
3. `update Movie set year = 1901 where director <> 'James Cameron';`
4. `update Rating set stars = stars - 1;`

III : PartieVérification de dépendances fonctionnelles

1. Avec la requête vue dans l'exercice 3 du TD2, vérifier que les dépendances de la question I.5 de l'exercice 2 de ce TP sont bien satisfaites.

2 Contraintes d'intégrité référentielle

Exercice 4 : contraintes d'intégrité référentielle

Oracle permet plusieurs types d'application des contraintes d'intégrité référentielle qu'il faut préciser au moment de la déclaration de la contrainte via `FOREIGN KEY`. Dans les exemples suivants `DEPT` est la *table parent* et `EMP` est la *table enfant*, c'est-à-dire respectivement la table *référencée* et celle sur laquelle *porte* la contrainte.

– Pour empêcher toute mise à jour ou suppression d'une clé de la table parent :

```
CREATE TABLE EMP (
  empno number(4),
  deptno number(2),
  FOREIGN KEY (deptno) REFERENCES DEPT (deptno));
```

– Quand une clé de la table parent est supprimée, tous les tuples de la table enfant qui dépendent de la valeur supprimée sont également supprimés :

```
CREATE TABLE EMP (
  empno number(4),
  deptno number(2),
  FOREIGN KEY (deptno) REFERENCES DEPT (deptno)
  ON DELETE CASCADE);
```

– Ne pas supprimer les tuples de la table enfant quand une clé de la table parent est supprimé. Les valeurs de la table enfant sont alors mises à `NULL` :

```
CREATE TABLE EMP (
  empno number(4),
  deptno number(2),
  FOREIGN KEY (deptno) REFERENCES DEPT (deptno)
  ON DELETE SET NULL);
```

Modifiez la définition d'une ou plusieurs tables pour prendre en compte les contraintes suivantes.

1. Intégrité référentielle entre `Rating.rID` et `Reviewer.rID` (suppression sur `Reviewers` : `SET NULL`).
2. Intégrité référentielle entre `Rating.mID` et `Movie.mID` (suppression sur `Movies` : `CASCADE`).
3. Traduire les contraintes des questions précédentes dans le formalisme des dépendances d'inclusion.

Exercice 5 : vérification des contraintes (suites)

Est-ce que les instructions suivantes génèrent des erreurs ou non ?

1. `insert into Rating values (209, 109, 3, '2001-01-01');`
2. `update Rating set rID = 209 where rID = 208;`
3. `update Rating set mID = mID + 1;`
4. `update Movie set mID = 109 where mID = 108;`
5. `update Movie set mID = 109 where mID = 102;`
6. `update Reviewer set rID = rID + 10;`
7. `delete from Reviewer where rID > 215;`
8. `delete from Movie where mID < 105;`
9. (†) Expliquer ce que fait la requête suivante (recherche de la documentation)

```
SELECT table_name,
       constraint_name,
       column_name
FROM USER_CONSTRAINTS NATURAL JOIN USER_CONS_COLUMNS
WHERE table_name in ('Movies', 'Rating', 'Reviewer');
```

3 Introduction à PL/SQL

Exercice 6 : introduction à PL/SQL

Il s'agit de pratiquer les principales constructions PL/SQL vues en cours. En cas d'erreur, utiliser les commandes `SHOW ERRORS;` et `SHOW WARNINGS;` pour obtenir des informations sur les dernières instructions executées

1. Écrire une procédure anonyme qui affiche `Hello World #` suivie de la valeur d'une variable `Nb`.
2. Remplacer la variable `Nb` de la question précédente par `&Nb`, que se passe-t-il ?
3. Avec l'instruction `SELECT ...INTO`, écrire une procédure anonyme qui affiche le nombre de lignes de la table `Movie`.
4. Reprendre la question précédente avec une boucle `FOR` pour afficher les lignes de `Film n` à `Film 1` où n est le nombre de films dans la relation.
5. Écrire une fonction paramétrée qui retourne le nombre d'évaluations effectuées par un relecteur dont l'identifiant est donné en paramètre.
6. Utiliser une requête SQL sur la table `DUAL` pour afficher le résultat de la fonction précédente pour un paramètre saisi par l'utilisateur.