LIF10 – FONDEMENTS DES BASES DE DONNÉES
TP3 – PL/SQL – groupes A & B1

Licence informatique – Automne 2013–2014

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

Résumé

Ce TP doit être réalisé seul ou en binôme dans le temps imparti de 2h et est à rendre *sous la forme d'un fichier `.sql` commenté* (pour le dépôt spiral il faut remplacer l'extension du fichier par *.txt*) contenant impérativement vos noms et prénoms et numéros d'étudiant sur Spiral. Le fichier doit être intitulé sous la forme suivante : **TP4-Nom1-Nom2.txt**.

La notation prendra en compte la qualité du code et celles des commentaires.

Exercice 1 : création de la base

Soit la base de données suivante où les clés primaires sont soulignées et où les clés étrangères sont précédées du symbole #. L'attribut *#id_cher_resp* désigne l'identifiant du responsable du projet et *nb_jour_sem* le nombre de jours par semaine qu'un chercheur consacre à un projet auquel il participe.

Equipe(*id_equipe*, *nom_equipe*)
Chercheur(*id_chercheur*, *nom_chercheur*, *nom_specialite*, *nom_universite*, *#id_equipe*)
Projet(*id_projet*, *nom_projet*, *#id_equipe*, *#id_cher_resp*)
Travaille(*#id_projet*, *#id_chercheur*, *nb_jour_sem*)

1. Créer cette base de donnée en SQL en prenant en compte les clés primaires et étrangères. Pour les clefs étrangères imposer que l'attribut référencés ne soit pas `NULL`. Utiliser les types de données suivants pour les attributs :
   – type `NUMBER(2)` pour les identifiants (préfixés par *id_*) et les nombres (préfixés par *nb_*),
   – type `VARCHAR2(20)` pour les chaînes de caractères des noms (préfixés par *nom_*).
2. Créer un jeu d'essai minimal pour les tests. Faire en sorte de bien avoir des cas limites comme une équipe sans chercheur (voir les questions de l'exercice 2).
3. Donner le message d'erreur Oracle qui survient lorsqu'une contrainte de clé étrangère est violée.

Exercice 2 : interrogation des données

Écrire les requêtes suivantes en SQL :

1. Quels sont les noms des projets et de leurs responsables pour lesquels le responsable ne travaille pas au projet.
2. Donner les noms des équipes qui ont des projets qui impliquent d'autres membres que ceux de l'équipe.
3. Donner le nom de projet et la somme du temps en jours que les chercheurs y consacrent. On souhaite en particulier voir la valeur 0 pour les projets auquel aucun chercheur ne travaille [1].

Exercice 3 : ajout des identifiants de spécialité

La gestion de spécialités d'après le nom n'est pas satisfaisante. On va créer une nouvelle table *Specialite* et s'appuyer sur des identifiants numériques.

1. Créer une nouvelle table. *Specialite*(*id_specialite*, *nom_specialite*) où *id_specialite* est clé et où *nom_specialite* est unique et non nul.
2. Afin de s'assurer de l'unicité des identifiants, on va créer une `SEQUENCE` qui permet de gérer l'auto-incrémentation. Créer pour cela la séquence à l'aide du code suivant :

1. La fonction `NVL(champ,val)` permet de remplacer un `NULL` de l'attribut `champ` par la valeur `val` fixée.

Algorithme 1: Mise-à-jour de la relation *Chercheur*

```
1 foreach chercheur c ∈ Chercheur do
2     if c.nom_specialite ∈ Specialite then
3         id := id_specialite trouvé dans Specialite
4     else
5         ajouter nom_specialite à Specialite
6         id := le nouvel id_specialite généré lors de l'ajout
7     mettre à jour c.id_specialite avec id
```

```
CREATE SEQUENCE specialite_seq
  MINVALUE 0
  START WITH 0
  INCREMENT BY 1
  NOCACHE;
```

3. Créer un déclencheur avant insertion sur la table *Specialite* qui remplace le *id_specialite* inséré par une nouvelle valeur générée par la séquence[2]. La requête `SELECT specialite_seq.nextval FROM dual;` permet de déterminer la nouvelle valeur de la séquence. Penser à utiliser la commande `SHOW ERRORS;` pour débugger le déclencheur.
4. Tester le déclencheur précédant en comparant l'état de la *Specialite* avant et après une insertion.
5. Ajouter un attribut numérique *id_specialite* à la table *Chercheur* avec une clé étrangère qui référence l'identifiant dans la table *Specialite*.
6. Créer une procédure utilisant un curseur dont le fonctionnement est donné en pseudo-code par l'algorithme 1.
7. Supprimer l'attribut *nom_specialite* désormais inutile dans la table *Chercheur*.

Exercice 4 : ajout des données calculées

1. Utiliser les commandes `ALTER TABLE` pour ajouter un attribut numérique *nb_chercheurs* à la relation *Specialite*.
2. Pour chaque spécialité, à l'aide d'une procédure, mettre à jour *nb_chercheurs* avec le nombre de chercheurs qui sont spécialistes de la dite spécialité.
3. La valeur de cet attribut doit être calculée dynamiquement. Pour cela, créer un déclencheur sur la relation *Chercheur* qui mettra à jour l'attribut *Specialite.nb_chercheurs* lors des ajouts, suppressions et modifications. Utiliser pour cela un déclencheur du type `AFTER INSERT OR UPDATE OR DELETE ON EACH ROW` et les prédicats `IF INSERTING`, `IF UPDATING('id_specialite')` et `IF DELETING`.
4. Vérifier que le déclencheur fonctionne bien.

2. En cas de popup intempestif demandant la valeur de `:New`, créer le trigger dans un nouveau fichier SQL que vous exécuterez en intégralité.