Université de Lorraine - Licence Informatique 3ème année - UE CMBD

TP 4 : PL/SQL (2/2)

Exercice 1 (Mise à jour à travers un curseur ; Trigger)

Considérer la table PRODUIT appartenant à l'utilisateur **smail5**.

SQL > DESC SMAIL5.PRODUIT

Name	Null ?	Type
PROD#	NOT NULL	NUMBER(38)
LIBELLE		VARCHAR2(20)
PU	NOT NULL	FLOAT(126)
TOTAL_VENTES		NUMBER(10)

1- Créer une copie de la table
CREATE TABLE toto AS (SELECT * FROM SMAIL5.PRODUIT)

2- Ecrire une procédure PL/SQL nommée **balai** qui supprime les **m** produits les moins vendus de la table PRODUIT. Réaliser la suppression à travers un curseur (CREATE CURSOR FOR UPDATE). La procédure aura **m** comme paramètre IN et retournera dans deux paramètres OUT le nombre de tuples effectivement supprimés et le nombre de tuples restants après la suppression.
3- Une fois la procédure mise au point, donner le droit d'exécution à un(e) camarade (GRANT EXECUTE ON nom_proc TO nom_camarade). La faire tester par ce(tte) camarade.

4- Définir un trigger nommé **SET_TV** qui se déclenche avant l'insertion d'un nouveau produit. Ce trigger teste si l'attribut TOTAL_VENTES est NULL et le positionne à zéro si c'est le cas. Tester que le trigger fonctionne dans au moins deux cas différents.

5- Voyez-vous une autre façon d'arriver au même résultat que ce qui est demandé à la question 4 ?

Exercice 2 (Paquetages ou Packages)

1- Copier le contenu de la table HR.JOBS dans une table JOBS.

2- Créer une table EMPLOYEES de même schéma que la tables HR.EMPLOYEES en **omettant** les attributs MANAGER_ID et DEPARTMENT_ID.

3- On souhaite créer un package comportant certaines opérations sur les employés. Pour cela, la spécification du package est fournie ci-après avec quelques commentaires. Créer le corps du package ACTIONS_EMP.

Afin de faciliter l'écriture des procédures et des fonctions publiques, il est possible (et conseillé) de définir des procédures et fonctions locales qui n'apparaissent pas dans la

spécification du package et qui ne seront visibles que dans le package. Par exemple, ici, une fonction booléenne SALAIRE_OK (JOB_ID VARCHAR2, SALARY NUMBER) pourrait permettre de tester la validité du nouveau salaire d'un employé par rapport à la grille des salaires.

```
CREATE PACKAGE ACTIONS_EMP AS

  /* Déclaration des types, curseurs et exceptions publics */

    TYPE EMP_REC_TYP IS RECORD (emp_id INT, salary REAL);

    CURSOR TRI_SALAIRES RETURN EMP_REC_TYP;

    SALAIRE_INVALIDE EXCEPTION;

  /* Déclaration des sous-programmes appelables à partir d'applications  */

 /* ajout d'un nouvel employé ; le numéro sera le successeur du numéro le
 plus élevé ; retourne le numéro de l'employé ajouté ; on suppose que la
 date d'embauche est la date du jour */

    FUNCTION EMBAUCHER (
       first_name  VARCHAR2,
       last_name  VARCHAR2,
       job_id    VARCHAR2,
       salary    REAL,
       commission_pct   REAL) RETURN NUMBER;

 /* suppression d'un employé */

    PROCEDURE LICENCIER (emp_id INT);

 /* augmentation du salaire d'un employé si le nouveau salaire de cet
 employé est OK par rapport à la grille des salaires contenue dans la table
 jobs (compris entre salaire min et salaire max). Sinon, l'exception
 SALAIRE_INVALIDE est levée  */

    PROCEDURE AUGMENTER_SALAIRE (emp_id INT, amount REAL);

 /* retourne le n-ième employé dans l'ordre décroissant des salaires */

    FUNCTION QUID_NIEME_SAL (n INT) RETURN EMP_REC_TYP;

 END emp_actions;
```

4- Tester le package créé en ajoutant quelques employés à la table EMPLOYEES. Augmenter le salaire de quelques employés dans les limites de la grille et en dehors des limites. Récupérer le 5ème employé dans l'ordre décroissant des salaires.

5- Donner à tous les utilisateurs le droit d'exécuter le package créé :

GRANT EXECUTE ON ACTIONS_EMP TO PUBLIC ;

6- Tester le package d'un camarade

EXECUTE MON_CAMARADE.ACTIONS_EMP.LICENCIER (…) ;

Commenter le résultat des tests.