DEFINITION DE TRAITEMENTS EN PL/SQL ET CREATION DE TRIGGERS

L'objet de ce TP est la définition de traitements sur les données d'une base ainsi que la définition de triggers, en utilisant le langage procédural d'ORACLE PL/SQL. Une attention particulière des gestions des exceptions, des commentaires sera faite.

Première étape : Gestion de valeurs

L'objectif de cette étape est d'écrire un programme PL/SQL permettant d'extraire de la base des données calculées et de les stocker dans une relation de travail pour les consulter ultérieurement. Plus précisément, on désire pour un abonné, dont le numéro sera donné par l'utilisateur à l'exécution du programme, déterminer le nombre total d'emprunts qu'il a effectués.

Pour cela, réalisez les opérations suivantes :

- Créer une relation AB_NB qui doit comporter seulement deux attributs : NUMERO dont la définition doit être compatible avec NUM_AB de ABONNE et NB défini comme un entier sur 3 positions.

- A l'aide de votre éditeur préféré, définissez le bloc PL/SQL permettant :
  - De stocker dans une variable le nombre d'emprunt d'un abonné ;
  - D'insérer le couple formé du numéro d'abonné et du nombre d'emprunts de cet abonné dans la relation de travail AB_NB ;

  Après exécution du programme, contrôlez les résultats obtenus en consultant l'extension de la relation AB_NB.

En phase de test, effectuez des **ROLLBACK**, pour défaire les opérations d'insertion réalisées.

- Terminez le programme en tenant compte de la possibilité de n'avoir aucun abonné pour le numéro spécifié (insertion dans AB_NB d'une valeur -1 comme nombre de prêt), de n'avoir aucun prêt pour l'abonné (insertion dans AB_NB de **NULL** comme nombre de prêt).

Enfin, après l'avoir testé, définissez le programme comme une transaction, i.e. en insérant comme première et dernière instruction du bloc, l'ordre **COMMIT**. Testez l'exécution et assurez-vous de l'impossibilité de défaire la transaction.

Deuxième étape : Fonction stockée

Il s'agit, en s'inspirant du programme précédent, de définir une fonction nb_pret qui accepte en paramètre d'entrée un numéro d'abonné et retourne le nombre de prêt de celui-ci.

- Créez la fonction nb_pret puis testez-la pour tous les abonnés de la base à l'aide d'une requête SQL.
- Modifiez le programme défini lors de la première étape afin qu'il utilise la fonction nb_pret.

Troisième étape : Gestion de curseurs

Il vous est demandé d'établir pour chaque abonné, le nombre d'emprunts réalisés pour chacune des catégories existantes.
Vous devez gérer tous les cas possibles : un abonné sans emprunt, un abonné sans emprunt pour une catégorie existante, une catégorie jamais empruntée.
Vous devez également traiter toutes les exceptions possibles.

Quatrième étape : Trigger de contrôle d'une CI simple

Pour un trigger, il ne faut pas définir des traitements pouvant être pris en charge par une contrainte explicite d'Oracle. Il ne s'agit ici que de tester les possibilités offertes par les triggers et de bien comprendre le modèle d'exécution des triggers par rapport aux contraintes statiques.

- A l'aide d'un trigger ***before***, il s'agit de traiter la contrainte sur la relation EXEMPLAIRE permettant d'interdire la suppression d'un exemplaire si son état est BON. Testez le trigger défini en tentant la suppression de l'exemplaire de n° 1012.

- A l'aide d'un trigger ***after***, il s'agit de traiter la contrainte sur la relation EMPRUNT permettant d'interdire l'emprunt d'un exemplaire si son état n'est pas BON. Testez le trigger défini en tentant d'emprunter l'exemplaire de n° 2673.

Cinquième étape : Trigger de mises à jour automatique

- Définissez le trigger permettant une affectation automatique de la date d'emprunt (en utilisant la fonction **SYSDATE** qui retourne la date du système) lors d'une insertion dans la relation EMPRUNT. De même, la date de retour sera calculée en ajoutant 21 jours à la date d'emprunt.