TP N°07 (semaine 48-49) : PLSQL – Procédures et fonctions

Exercice N°1 : Procédure – Appel de procédure

Une procédure stockée est un bloc de code PL/SQL nommé et défini par l'utilisateur.
Elle est composée d'instructions compilées et enregistrées dans la base de données.
Elle est paramétrable afin d'en faciliter la réutilisation.
Elle peut être exécutée à partir des applications ou d'autres procédures stockées.
Elle comporte, outre des ordres SQL, des instructions de langage de troisième génération (branchement conditionnel, instructions de répétition, affectations...).

a) Ecrire la procédure PL/SQL ***clientville (p_ville IN...)*** qui, selon la ville *p_ville* passée en paramètre, affiche la liste des clients de cette ville (s'il y en a).

Tester la procédure en dehors d'un bloc PL/SQL grâce à la commande EXECUTE.

Ex. : Ville :REIMS

DUBOIS Paul
…
Le nombre de clients REIMS est : 4

b) Utiliser la procédure dans un nouveau bloc PL/SQL (anonyme) pour chaque ville de la **Marne (**trouvée dans la table ADRESSE)

Exercice N°2 : Procédure – Appel de procédure

a) Ecrire la procédure PL/SQL ***agt_chb (p_num IN..., p_nom OUT..., p_prenom OUT...)*** qui selon l'identifiant *p_num* d'une chambre (IN), renvoie dans les paramètres de sortie (OUT) le nom et le prénom de l'agent en charge de cette chambre.

Si la chambre n'existe pas on déclenche une erreur (RAISE_APPLICATION_ERROR) associée au message 'ERREUR : Chambre inexistante'.

b) Ecrire un bloc PL/SQL pour tester la procédure sur une chambre dont l'identifiant sera saisi par l'utilisateur et qui affiche les résultats correspondants.
Utiliser PRAGMA EXCEPTION_INIT pour gérer le cas d'une chambre inexistante et afficher le message d'erreur correspondant.

Ex. : BOUSSEL Annie est en charge de la chambre 15
PEZIN Franck est en charge de la chambre 20
ERREUR : Chambre 25 inexistante

c) Exécuter la procédure dans un bloc PL/SQL pour toutes les chambres réservées (présentes dans la table planning) à un étage et une date saisis par l'utilisateur.
Gérer l'erreur utilisateur d'un étage incorrect (« RDC », « 1er » ou « 2e ») et l'erreur de date (date incorrecte)

Ex. : *Pour l'étage RDC le 11/11/2007 on affichera :*

-------PERSONNEL ATTENDU le 11/11/2007 A l'ETAGE : RDC--------
BOUSSEL Annie pour la chambre 1
SAUVAGE Laurent pour la chambre 3
BOUSSEL Annie pour la chambre 4

Pour l'étage 3e le 11/11/2007 on affichera :

Ce n'est pas un etage valide

Pour l'étage RDC le 11/11/2010 on affichera :

-------PERSONNEL ATTENDU le 11/11/2010 A l'ETAGE : RDC--------
Pas de chambre réservée le 11/11/10 à l'étage RDC

Exercice N°3 : Fonctions PL/SQL

Une fonction est un sous-programme contenant un certain nombre d'actions et qui retourne un résultat unique affecté à son nom. Elle est semblable à une procédure stockée, la seule différence étant qu'elle retourne toujours une valeur.

Elles permettent d'augmenter l'indépendance des données en faisant le calcul côté serveur Oracle au lieu de le faire côté client (application).

a) Créer une fonction **idPlus1( )** qui retourne l'identifiant max + 1 de la table Client.

Tester la fonction en l'utilisant dans un SELECT ... FROM DUAL ;
Tester la fonction en insérant un nouveau client.

Tit_code	Cli_nom	Cli_prenom
M.	TUTIN	Michel

Vérifier en affichant les clients dont le cli_id est supérieur à 100

b) Créer une fonction **EstEnRegle(p_cli_id...)** qui renvoie **vrai** ou **faux** selon que le client a une facture impayée ou non (facture qui n'a pas de date de paiement).

c) Créer une fonction **Existe(p_cli_id ...)** qui renvoie vrai ou faux selon que le client existe ou non.

d) Tester les fonctions en créant un bloc PLSQL qui affiche un message selon que le client dont le numéro est saisi par l'utilisateur existe ou non, et est en règle ou non.

Ex. : pour le client 99 → « Le client 99 est en règle »
Pour le client 69 → « Le client 69 n'a pas payé ses factures »
Pour le client 200 → « Le client n'existe pas ».

Exercice N°4 :

a) Créer la fonction ***table_existe (nom_table IN VARCHAR2)*** qui renvoie VRAI (un booléen) lorsque la table dont le nom passé en paramètre existe et qui renvoie FAUX sinon.

Exemples : table_existe ('CLIENT') → renvoie VRAI
table_existe('TOTO') → renvoie FAUX

Utiliser une requête sur la table **USER_TABLES** (colonne **TABLE_NAME**)

b) Ecrire un bloc PL/SQL qui selon le nom de la table saisi par l'utilisateur affiche un message correspondant :

- La table CLIENT existe
- La table TOTO n'existe pas

Exercice N°5 : (facultatif)

a) Créer la procédure ***supprime_table(nom_table IN VARCHAR2)*** qui supprime la table dont le nom est passé en paramètre, si elle existe (utiliser la fonction précédemment créée).

Exemples :
```
CREATE TABLE client_reims AS
        (SELECT cli_nom, cli_prenom, adr_cp, adr_ville
            FROM client c, adresse a
            WHERE c.cli_id = a.cli_id AND  adr_ville= 'REIMS') ;
```

supprime_table ('CLIENT_REIMS') → la table CLIENT_ REIMS est supprimée.
supprime_table('TOTO') → la table TOTO n'existe pas.

La commande EXECUTE IMMEDIATE va permettre d'exécuter un ordre non supporté par le SQL statique (CREATE, ALTER, DROP, GRANT, REVOKE, ...) dans un bloc PL/SQL.

```
EXECUTE IMMEDIATE chaîne_dynamique
[INTO {variable, … |enregistrement}]
[USING [IN|OUT|IN OUT] argument …]
[{RETURNING|RETURN} INTO argument, …]
```

Où **chaîne_dynamique** : ordre SQL ou bloc PL/SQL
variable : var. pour stocker la valeur d'une colonne
enregistrement : var. structurée pour stocker la valeur d'une ligne
argument : valeurs passées à l'ordre SQL ou au bloc PL/SQL ,de type autre que BOOLEAN.

Exemples :
```
EXECUTE IMMEDIATE 'CREATE TABLE clients
                   (nocli number(6), nom varchar2(10), ca number(8,2))';
```

b) Créer la fonction ***nb_rows (nom_table VARCHAR2)*** qui renvoie le nombre de lignes de la table dont le nom est passé en paramètre.

Exemples : nb_rows ('CLIENT') → renvoie 103
nb_rows('TOTO') → renvoie 0 (car la table n'existe pas)