masquer=1

TME 10 : PL/SQL

LE SGBD ORACLE

Dans ce TME nous allons utiliser le SGBD Oracle. Suivez les instructions du document Oracle avec SQLWorkbench pour vous connecter au serveur Oracle.

REQUÊTES A RESULTAT UNIQUE

Oracle peut exécuter des commandes SQL (SELECT, INSERT, DELETE UPDATE) et des *blocs PL/SQL*

Une suite de commandes contenant un ou plusieurs blocs PL/SQL doit être terminée par un /.

Bloc PL/SQL

Un bloc PL/SQL a la structure suivante (les [..] encadrant les éléments optionnels) :

```
[ DECLARE
   -- déclaration de variables ou d'exceptions ]
  BEGIN
   -- instructions PL/SQL
[ EXCEPTION
   -- actions déclenchées par les évènements répertoriés]
  END;
  /
```

Remarque : Une suite de commandes contenant un ou plusieurs blocs PL/SQL doit être terminée par un / pour être compilé.

Variables

1) Variables SQL*Plus

La commande ACCEPT de SQL*Plus permet de définir une variable et d'afficher un texte avant sa saisie. Cette commande doit figurer à l'extérieur d'un bloc PL/SQL. La valeur de la variable peut ensuite être utilisée partout, en ajoutant un & en préfixe à son nom. Par défaut, la variable est de type CHAR .

2) Variables PL/SQL

Les autres variables utilisées dans un bloc PL/SQL doivent être déclarées dans la section DECLARE de ce bloc. Contrairement aux variables SQL*Plus, elles sont ensuite directement désignées par leur nom, sans préfixe &. Elles peuvent avoir, pour type, n'importe quel type du langage SQL. L'attribut %TYPE permet de faire référence au type d'une colonne donnée.

3) Assignation de variables PL/SQL

On utilise l'opérateur := pour assigner une valeur à une variable, lors de sa déclaration ou dans une instruction de la section BEGIN.

Fonctions de conversion

Les fonctions *to_char* et *to_number* permettent de convertir une valeur numérique en chaîne de caractères, et inversement. On peut également appliquer des *fonctions de conversion* pour modifier les valeurs de variables ; Par exemple les fonctions UPPER et LOWER transforment leur arguments (des chaînes de caractères) en majuscule ou minuscule.

Opérateurs

L'opérateur || concatène deux chaînes de caractères.

Instructions PL/SQL :

Une e*xpression SQL* est une instruction PL/SQL. Par exemple, on peut utiliser l'ordre UPDATE pour modifier la base de données. On suppose que chaque requête SQL retourne au maximum un nuplet. La valeur de cette nuplet peut être copiée dans des variables déclarées dans la section DECLARE. Par exemple :

```
select Jour, Heure, Salle
into JourI, HeureI, SalleI
from TD
where niveau=:new.niveau and UE=:new.UE and NoTD=:new.NoTD;
```

Il est possible d'utiliser les structures de contrôle suivants :

- `if (<condition>) then <bloc_psql1> [ else <bloc_plsql2> ] end if;`
- `loop <bloc_psql> end loop;`
- `for <var> in <seq> loop <bloc_psql> end loop ;`
- `while <condition> loop <bloc_psql> end loop ;`

Les commandes `exit` et `exit when <cond>` permettent de sortir d'une boucle.

Package dbms_output

Ce package fournit la fonction *put_line* qui permet d'afficher un texte à l'écran, si l'option SERVEROUTPUT de *sqlplus* est positionnée. **Remarque** : il n'est pas possible d'utiliser ce package avec SQLWorkbench (nous allons écrire des bloc qui retournent des résultats de requêtes).

Exceptions définies par l'utilisateur

Une telle exception doit être déclarée dans la section DECLARE du bloc PL/SQL concerné. Lorsqu'elle est levée (commande *raise*), *sqlplus* exécute l'instruction prévue dans la section EXCEPTION et met fin à l'exécution du bloc concerné.

Procédures et fonctions stockées

On crée (ou modifie) une procédure par la commande :

```
CREATE OR REPLACE PROCEDURE Nom_Proc
     (Paramètre1 type1, … , Paramètrek typek)
IS
Bloc PL/SQL
/
```

Les types possibles des paramètres sont ceux des variables PL/SQL; on peut notamment utiliser l'attribut %TYPE. Le bloc PL/SQL, constituant le corps de la procédure, peut commencer par une section de déclaration de variables, mais sans le mot réservé DECLARE.

Les fonctions permettent de retourner des résultats. On crée (ou modifie) une fonction par la commande :

```
CREATE OR REPLACE FUNCTION Nom_Fonction RETURN type_resultat
     (Paramètre1 type1, … , Paramètrek typek)
IS
Bloc PL/SQL (contenant les instructions RETURN)
/
```

A l'exécution de cette commande, la procédure est compilée et stockée dans la base. Elle est alors utilisable par différents programmes, qui pourront l'appeler par la commande:

execute *Nom_Proc*(*Valeur1*, …, *Valeurk*) ;

Si la compilation de la procédure s'est terminée sur erreur, on peut lancer la commande SQL*Plus *show errors* pour obtenir des précisions.

Exercices

Les exercices suivants portent sur les tables GAIN, JOUEUR et RENCONTRE de la base TENNIS, implémentée dans la base commune *oracle*. Chaque exercice donne lieu à la création d'une procédure et d'un programme de test de cette procédure. On pourra rassembler les commandes de création des procédures, dans un même fichier, en éditant ce fichier sous Xemacs, et en lançant la création de chaque nouvelle procédure par sélection du code correspondant. Chaque programme de test fera l'objet d'un fichier séparé ; on l'exécutera par la commande *@Nom_de_Fichier*, sous SQL*Plus.

Préparation :

1. *Lancez SQL Workbench dans le repertoire workbench_avec_driver*
2. *Connectez vous à Oracle en suivant les instructions du document Oracle avec SQLWorkbench*
3. *Chargez le script SQL suivant Oracle en tapant la commande suivante dans l'éditeur SQL WorkBench (vous pouvez ouvrir plusieurs onglets):*

```
@tennis
```

1. Créer la fonction moyprime. Copier-coller le code suivant dans l'éditeur :

```
CREATE OR REPLACE FUNCTION moyprime(
  P_lieutournoi GAIN.lieutournoi%TYPE,
  P_annee GAIN.annee%TYPE)
RETURN VARCHAR2 IS
V_moyenne GAIN.prime%TYPE;
E_fin EXCEPTION;
BEGIN
  select AVG(prime) into V_moyenne
  from GAIN
  where lieutournoi=P_lieutournoi and annee=P_annee;
  IF V_moyenne IS NULL THEN RAISE E_fin; END IF;
  RETURN(P_lieutournoi||' '||to_char(P_annee)||': '
                          ||to_char(V_moyenne));
  EXCEPTION
  WHEN E_fin THEN RETURN(P_lieutournoi||' '||to_char(P_annee)
                        ||' non répertorié');
END;
/
```

Exécutez les deux requêtes suivantes et expliquez le résultat (la table dual contient un seul nuplet et permet d'effectuer un appel de fonction à travers une requêtes SQL

```
select distinct moyprime(lieutournoi,annee) as reponse
from GAIN;

select moyprime('Wimbledon',1990) as reponse
from dual;
```

Ajouter un commentaire au code de création de la procédure, pour indiquer ce qu'elle fait (-- *Affiche* ...).

2. Définir une fonction moyprime2 produisant les mêmes effets que la procédure précédente, sans utiliser d'exception E_fin. La tester avec un script moyprime2.sql.

REQUÊTES A RÉSULTAT MULTIPLE

Rappels

Boucle

Une boucle PL/SQL est encadrée par les mots *loop* et *end loop*. Précédée de la clause *for*, elle est exécutée un certain nombre de fois ; précédée de la clause *while*, elle est exécutée tant qu'une condition reste satisfaite ; en l'absence de clause *for* ou *while*, elle s'arrête à la rencontre d'une instruction *exit*.

On peut placer une boucle à l'intérieur d'une autre boucle. Il est alors préférable de nommer chaque boucle, en plaçant une étiquette juste avant le début de la boucle (<<*nom_boucle*>> loop …. exit *nom_boucle* when… …. end *nom_boucle* ;).

Curseur

Lorsqu'une requête est susceptible de produire plusieurs lignes en résultat, on utilise un curseur pour accéder à chacune de ces lignes. Le curseur doit être défini dans la section DECLARE du bloc où il est utilisé. L'instruction *open* positionne le curseur sur la première ligne du résultat. L'instruction *fetch* permet de recopier les valeurs des attributs de la ligne courante dans des variables, et positionne le curseur sur la ligne suivante. L'instruction *close* libère les ressources allouées au curseur.

Un curseur possède des attributs :

%FOUND a la valeur vrai si une instruction *fetch* a effectivement pu lire une ligne ;

%NOTFOUND a la valeur vrai dans le cas contraire ;

%ROWCOUNT comptabilise le nombre de lignes effectivement lues par une série d'instructions *fetch*.

Un curseur peut être défini avec des paramètres dont la valeur est fixée lors de l'ouverture:

cursor *nom_curseur* (*param1 type1*, *param2 type2,* ...) is … ;
open *nom_curseur* (*valeur1*, *valeur2*, …) ;

Fonctions de manipulation de chaînes de caractères

La fonction *length* retourne le nombre de caractères d'une chaîne.
La fonction *rpad* répète un motif donné, à la droite d'une chaîne, pour obtenir une chaîne de longueur donnée; la fonction *lpad* répète le motif à gauche de la chaîne.

Exercices

3. Expérimenter la fonction et la requête de test suivants ; puis ajouter un commentaire indiquant ce que fait la procédure.

```
CREATE or REPLACE FUNCTION maxprime
```

```
( P_Annee1 GAIN.annee%TYPE, P_Annee2 GAIN.annee%TYPE )
RETURN CLOB
IS
V_Nom Joueur.nom%type;
V_MaxPrime Gain.Prime%type;
V_reponse CLOB;
cursor C_MaxPrime is
select nom, max(prime)
from GAIN, JOUEUR
where annee between P_Annee1 and P_Annee2
  and gain.NuJoueur=joueur.NuJoueur
group by nom;
BEGIN
open C_MaxPrime;
V_reponse:='';
loop
  fetch C_MaxPrime into V_Nom, V_MaxPrime;
  exit when C_MaxPrime%NOTFOUND;
  if C_Maxprime%ROWCOUNT=1 then
    V_reponse:=V_reponse||'Plus forte prime gagnée entre '
        ||P_Annee1||' et '||P_Annee2||':';
  end if;
  V_reponse:=V_reponse||chr(10)
        ||rpad(V_Nom,14,'.')||lpad(to_char(V_MaxPrime),8,'.')||' F';
end loop;
if C_MaxPrime%ROWCOUNT=0
 then V_reponse:='Aucun tournoi n''est répertorié entre ces dates';
end if;
close C_MaxPrime;
return V_reponse;
END;
/

-- @WbOptimizeRowHeight lines=100
select maxprime(1990,1994) as réponse from dual;

select maxprime(1970,1975) as reponse from dual;
```

4. Modifier la procédure précédente de façon à obtenir une fonction maxprime2 admettant en paramètre le nom d'un sponsor, et qui :
   - détermine la première et la dernière année où ce sponsor a figuré dans un tournoi,
   – puis affiche, pour chaque joueur, la plus forte prime qu'il a touchée durant cette période (les joueurs pris en compte sont ceux qui ont participé à, au moins, un tournoi dans la période considérée, qu'ils aient, ou non, été sponsorisés par le sponsor indiqué). La donnée d'un sponsor inconnu dans la base provoque l'arrêt immédiat du programme, par levée d'une exception.

```
-- @WbOptimizeRowHeight lines=100
select maxprime2('Peugeot') as réponse from dual;

RÉPONSE
Plus forte prime donnée par Peugeot
entre 1989 et 1994:
FORGET.........400000 F
SAMPRAS........1400000 F
LECONTE.........350000 F
FLEURIAN........600000 F
```

```
-- @WbOptimizeRowHeight lines=100
select maxprime2('Reebok') as réponse from dual;
```

```
RÉPONSE
Plus forte prime donnée par Reebok entre
1993 et 1994:
PIERCE.........350000 F
FORGET.........600000 F
SAMPRAS........1800000 F
LARSSON.........800000 F
GRAF............400000 F
LECONTE........1000000 F
```

5. Ecrire une fonction *jouspon*, qui affiche les noms des joueurs qui ont été sponsorisés par le sponsor dont le nom est transmis en paramètre. Tester la procédure avec un script JouSpon.sql dont l'exécution donnera, par exemple :

```
-- @WbOptimizeRowHeight lines=100
select jouspon('Reebok') as réponse from dual;
```

```
RÉPONSE
Joueurs sponsorisés
par Reebok:
PIERCE
FORGET
SAMPRAS
LARSSON
GRAF
LECONTE
```

```
-- @WbOptimizeRowHeight lines=100
select jouspon('Peugeot') as réponse from dual;
```

```
RÉPONSE
Joueurs sponsorisés
par Peugeot:
FORGET
SAMPRAS
LECONTE
FLEURIAN
```

```
-- @WbOptimizeRowHeight lines=100
select jouspon('Head') as réponse from dual;
```

```
RÉPONSE
Head n'a sponsorisé
```

```
personne
```

6. Compléter la procédure précédente, de façon à ce qu'elle indique, pour chaque joueur figurant en résultat, les tournois qu'il a gagnés (avec le sponsor donné, ou un autre) ; les joueurs n'ayant gagné aucun tournoi apparaîtront, dans la liste, avec leur seul nom, comme auparavant. On obtiendra ainsi, par exemple :

```
-- @WbOptimizeRowHeight lines=100
select jouspon2('Reebok') as réponse from dual;

RÉPONSE
Joueurs sponsorisés par Reebok:
GRAF vainqueur de Wimbledon et de Wimbledon
1992
LARSSON n'a pas gagné de tournoi
LECONTE n'a pas gagné de tournoi
PIERCE n'a pas gagné de tournoi
FORGET n'a pas gagné de tournoi
SAMPRAS vainqueur de Roland Garros et de
Wimbledon 1993


-- @WbOptimizeRowHeight lines=100
select jouspon2('Peugeot') as réponse from dual;

RÉPONSE
Joueurs sponsorisés par Peugeot:
LECONTE n'a pas gagné de tournoi
FORGET n'a pas gagné de tournoi
FLEURIAN n'a pas gagné de tournoi
SAMPRAS vainqueur de Roland Garros et de
Wimbledon 1993



-- @WbOptimizeRowHeight lines=100
select jouspon2('Head') as réponse from dual;

RÉPONSE
Head n'a sponsorisé
personne
```