TME 10 : PL/SQL

1. REQUÊTES A RESULTAT UNIQUE

1.1 Rappels

1.1.1 SQL*Plus

L'interpréteur *sqlplus* d'Oracle peut exécuter trois sortes de commandes:
- des commandes SQL;
- des blocs PL/SQL;
- des commandes SQL*Plus permettant de positionner des options, de modifier la présentation des résultats, d'exécuter des fichiers de commandes...

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

1.1.2 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;
```

1.1.3 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.

1.1.4 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.

1.1.5 Fonctions de conversion

Les fonctions *to_char* et *to_number* permettent de convertir une valeur numérique en chaîne de caractères, et inversement.

1.1.6 Opérateurs

L'opérateur || concatène deux chaînes de caractères.

1.1.7 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.

1.1.8 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.

1.1.9 Procédures 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.

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.

1.2. 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.

1.2.1 Créer la procédure PROCmoyprime. Copier-coller le code suivant dans l'éditeur :

```
CREATE or REPLACE PROCEDURE PROCmoyprime
( P_lieutournoi GAIN.lieutournoi%TYPE, P_annee GAIN.annee%TYPE )

IS

V_moyenne GAIN.prime%TYPE;
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 FIN;
 end if;

dbms_output.put_line(P_lieutournoi||' '||to_char(P_annee)||': '
                                        ||to_char(V_moyenne));
EXCEPTION
when FIN then dbms_output.put_line('Tournoi non répertorié');

END;
/
```

Puis expérimenter le script suivant, pour divers lieux et diverses années de tournoi :

```
-- fichier moyprime.sql
```

```
-- utilise la procedure PROCmoyprime

ACCEPT A_lieutournoi PROMPT "Donnez le lieu du tournoi: "
ACCEPT A_annee NUMBER PROMPT "Donnez l'année: "
SET SERVEROUTPUT ON
SET VERIFY OFF
execute PROCmoyprime('&A_lieutournoi', &A_annee);
```

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

1.2.2 Définir une procédure PROCmoyprime2 produisant les mêmes effets que la procédure précédente, sans utiliser d'exception. La tester avec un script moyprime2.sql.

I

2. REQUÊTES A RÉSULTAT MULTIPLE

2.1. Rappels

2.1.1 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* ;).

2.1.2 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*, …) ;

2.1.3 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.

2.2. Exercices

2.2.1 Expérimenter la procédure et le programme de test suivants ; puis ajouter un commentaire indiquant ce que fait la procédure.

```
CREATE or REPLACE PROCEDURE PROCmaxprime
( P_Annee1 GAIN.annee%TYPE, P_Annee2 GAIN.annee%TYPE )

IS

V_Nom Joueur.nom%type;
V_MaxPrime Gain.Prime%type;

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;

loop
fetch C_MaxPrime into V_Nom, V_MaxPrime;
exit when C_MaxPrime%NOTFOUND;
if C_Maxprime%ROWCOUNT=1
 then dbms_output.put_line('Plus forte prime gagnée par:');
end if;
dbms_output.put_line(rpad(V_Nom,14,'.')||
                     lpad(to_char(V_MaxPrime),8,'.')||' F');
end loop;

if C_MaxPrime%ROWCOUNT=0
 then dbms_output.put_line
                  ('Aucun tournoi n''est répertorié entre ces dates');
end if;

close C_MaxPrime;

END;

/


-- Fichier MaxPrime.sql

SET SERVEROUTPUT ON
SET VERIFY OFF
ACCEPT A_Annee1 NUMBER PROMPT 'Année de départ: '
ACCEPT A_Annee2 NUMBER PROMPT 'Dernière année: '
execute PROCmaxprime(&A_Annee1, &A_Annee2);
```

2.2.2 Modifier la procédure précédente de façon à obtenir une procédure PROCmaxprime2 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.

2.2.3 Ecrire une procédure PROCjouspon, 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 :

```
SQL> @JouSpon
Nom du sponsor: Reebok
Joueurs sponsorisés par Reebok:
GRAF
PIERCE
LARSSON
LECONTE
FORGET
SAMPRAS

Procédure PL/SQL terminée avec succès.

SQL> @JouSpon
Nom du sponsor: Toto
Toto n'a sponsorisé personne

Procédure PL/SQL terminée avec succès.
```

2.2.4 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 :

```
SQL> @JouSpon2
Nom du sponsor: Reebok
Joueurs sponsorisés par Reebok:
GRAF vainqueur de Wimbledon 1989
..............de Wimbledon 1992
PIERCE
LARSSON
LECONTE
FORGET
SAMPRAS vainqueur de Roland Garros 1994
..................de Wimbledon 1993

Procédure PL/SQL terminée avec succès.
```