Université Mohammed V - Rabat

Ecole Mohammadia d'Ingénieurs

Filière Réseaux et Télécom

Bases de Données

Programmation PL/SQL

Mr N.EL FADDOULI

2022-2023

Plan

- Introduction
- Bloc PL/SQL
- Déclaration des variable
- Structure de contrôle
- Curseurs
- Les exceptions
- Les fonctions et procédures
- Les packages
- Les triggers

Concepts de base

- Introduction
- Bloc PL/SQL
- Déclaration des variable
- Structure de contrôle

Introduction

- Le PL/SQL est un langage procédural d'ORACLE.
- L'intérêt du PL/SQL est de pouvoir dans un même traitement combiner la puissance des **instructions SQL** et la souplesse d'un **langage procédural**.
- Le fonctionnement de PL/SQL est basé sur l'**interprétation** d'un **bloc** de commandes.
- Dans l'environnement SQL, les ordres du langage sont transmis et exécutés les uns à la suite des autres.
- Dans l'environnement PL/SQL, les ordres SQL et PL/SQL sont regroupés en blocs; un bloc ne demande qu'un seul transfert pour l'exécution de l'ensemble des commandes contenues dans le bloc.

Bloc PL/SQL

- Un script PL/SQL est constitué d'un ou plusieurs blocs qui constituent le module d'exécution. Il y a trois parties ou sections distinctes dans un bloc: DECLARE, BEGIN et EXCEPTION.

DECLARE

Déclaration des variables locales au bloc, constantes, types (record), exceptions, curseurs

BEGIN

Commandes exécutables: Instructions SQL et PL/SQL. Possibilité de ***blocs imbriqués****.*

EXECPTION

Traitement des erreurs

END ;

Bloc PL/SQL

- Un bloc PL/SQL peut contenir:
  - Toute instruction du LMD (SELECT, INSERT, UPDATE, DELETE)
  - Les commandes de gestion des transactions (COMMIT, ROLLBACK, SAVEPOINT)
- Les sections **DECLARE** et **EXCEPTION** sont optionnelles.
- Chaque instruction se termine par **;**
- Les blocs peuvent être imbriqués:
  - Les sous blocs ont la même structure que les blocs.

Déclaration des variables

- La partie déclarative dans un bloc PL/SQL, peut comporter 3 types de déclarations:
  - **Déclarations des variables, constantes**
  - Déclaration des curseurs
  - Déclaration de records
  - Déclaration des exceptions
- **Variables ou constantes**

  Nom-de-variable* [CONSTANT]** ***Type **[NOT NULL] [[:= | DEFAULT]** ***Expr*** **];**

  Expr : une constante ou un calcul faisant éventuellement référence à une variable précédemment déclarée. L'initialisation est **obligatoire** si **CONSTANT** ou **NOT NULL** sont utilisés.

Exemple: DECLARE

```
Nom_duclient    char (30);        -- la valeur par défaut est NULL
X               number:=12;
Y               number:=X*X;      -- L'ordre de déclaration est important.
PI   constant   number(7,5)  DEFAULT  3.14159 ;
```


Déclaration des variables

- **Type de variables:**
  - **Scalaires** : reçoivent une seule valeur
  - Composées: permettent de définir des groupes de champs et de les manipuler dans des blocs PL/SQL (tels que les **records**)
  - Références
  - LOB (Large Object)
- **Types de données scalaires:**
  - Nombres:
    X **Number(7,2)**; -- *7 chiffres dont 2 après la virgule*
    Y **Number**; -- *par défaut: 38 chiffres*
    Z **Number(5)**;
  - Caractères:
    S **Char(20)**; -- *Chaîne de taille fixe*
    T **Varchar2(30)**; -- *Chaîne de taille variable*
  - Dates:
    D **Date** NOT NULL := **SYSDATE**; -- *date système*
  - Booléens:
    B **Boolean** : = True;

Déclaration des variables

- **Variables ayant le même type qu'une colonne ou le même type qu'une autre variable**

  Nom-de-variable ***Nom_table.Nom-colonne%type*** ;

  ou

  Nom-de-variable1 ***Nom-de-variable2%type*** ;

  Exemple: DECLARE

  A Emp.EmpNo%type;
  X number (10,3);
  Y X%type;

- **Assignation (Affectation)**
  - Par l'opérateur d'affectation : **:=**
  - Par la clause : **Select….Into**
  - Par le traitement d'un **curseur** dans la section **BEGIN**.

Déclaration des variables

- **Assignation** *(Suite)*
  - Par l'opérateur d'affectation : **:=**

  Nom_variable **:=** ***expression*** **;**

  Exemple :

```
DECLARE
        X       Number;
        Y       Number  Not Null := 3;
        Z       Number;
BEGIN

X := 0 ;
Y := ( X+5) * Y ;
Y := Z;  -- ORA-06502: PL/SQL : erreur numérique ou erreur sur une valeur
END;
```

Parce que Z=NULL

Déclaration des variables

- **Assignation** *(Suite)*
  - Par la clause **SELECT….INTO : La requête Select doit retourner UNE seule valeur**

Exemple:

```
DECLARE
        Nom    Dept.Dname%type ;
BEGIN
        SELECT Dname  INTO  Nom
        FROM  Dept  WHERE  DeptNo=20 ;
END;
```


Déclaration des variables

Exemple:

```
DECLARE
      a number;
      b varchar2(20);
BEGIN
    b:="Salut";
    a:=22;
    DECLARE
          a   number;
    BEGIN
          a:=33;    b:="Bonjour";
          dbms_output.putline(a || b);
    END;
    dbms_output.put_line(a || b)
END;
```

Une variable est visible dans le bloc où elle est déclarée et dans tous ses sous-blocs.

Si une variable est déclarée dans un premier bloc et aussi dans un sous bloc, la variable du bloc supérieur n'est plus visible dans le sous-bloc

Déclaration des variables

- **Variables Hotes SQL*PLUS**

  - Ce sont des variables qui sont accessibles par n'importe quel bloc PL/SQL.

  - Elles sont déclarées par la commande **VARIABLE** ou **VAR** comme suit:

    SQL > **Variable** *Nom_Variable_Hote* ***Type***;

  Exemple: **SQL> Variable** Total Number(5,2);

  - Ces variables sont accessibles dans la partie **BEGIN** d'un bloc PL/SQL comme suit:

    :*Nom_Variable*

  Exemple: **:** Total := 2345;

  - Une variable hôte peut être affichée à partir de SQL*PLUS ave la commande **PRINT** comme suit: **SQL> PRINT** *Nom_Variable*

Affichage de résultat

- Positionner **SERVEROUTPUT** à **ON**:

  SQL> SET **ServerOutput** On

- Utiliser la fonction **PUT_LINE** du package: **DBMS_OUTPUT**

  DBMS_OUTPUT. PUT_LINE (*expression*);

Exemple:

```
Declare
        f   varchar2(20);
        a   number(5);
Begin
        a :=34;   f := 'Salut tout le monde';
dbms_output.put_line(f ||' '||a);   -- Affichage d'une chaîne et d'un entier.
dbms_output.put_line(:Total);       -- Affichage de la variable hôte Total
End;
/
```

Variables de Substitution

- Passage de paramètres en entrée d'un bloc PL/SQL
- Utilisation de l'opérateur de substitution: **&**

Exemple:

```
Declare
        f   varchar2(20);   a   number(3);
Begin
        f := '&nom';
        a:=&age;
        dbms_output.put_line ('Votre Nom ' ||f|| ' Vous avez  '||a||'  ans');
End;
   /
```

N.B: - Le symbole **nom** (ou **age**) n'est pas une variable à déclarer. Il est utilisé par Oracle pour afficher un message demandant la saisie d'une valeur.

- Pour empêcher l'affichage automatique des valeurs avant et après la saisie dans SQLPLUS, on utilise: **SET VERIFY OFF**

Variables de Substitution

- On utilise la directive SQL*Plus **ACCEPT** si on veut afficher un message d'invite personnalisé.

Exemple: Le contenu du fichier **d:\tpbd\prompt.sql**

```
ACCEPT a  char     PROMPT "Entrez le nom de l'employé: " ;
ACCEPT b  number   PROMPT ' Entrez l''âge de l''employé: ' ;
      Declare
              nom  varchar(20);  age number(3);
      Begin
        nom:='&a';   age:=&b ;
        dbms_output.put_line ('Votre Nom ' ||nom||'    Vous avez '||   age   ||' ans');
        -- …….
      End;
       /
```

Depuis l'invite de SQLPLUS: SQL> **start** 'd:\tpbd\prompt.sql'

Structures de contrôles

Structure alternative

- **IF …. THEN…. END IF**
- **IF …. THEN…. ELSE … END IF**
- **IF ….. THEN ELSIF ….**

Exemples:

1. IF *condition* THEN
   sequence_insts;
   END IF;

2. IF *condition* THEN
   sequence_insts1;
   ELSE
   sequence_insts2;
   END IF;

Structures alternative

Structure alternative

Exemples (*suite*)

3. IF *condition1* THEN
   sequence_insts1;
   ELSIF *condition2* THEN
   sequence_insts2;
   ELSIF *condition3* THEN
   sequence_insts3;
   ELSE
   sequence_insts3;
   END IF;

```
IF Moyenne< 10 THEN
        Resultat := 'Echec';
ELSIF Moyenne< 12 THEN
        Resultat := 'Passable';
ELSIF Moyenne< 14 THEN
        Resultat := 'Assez Bien';
ELSIF Moyenne< 15 THEN
        Resultat := 'Bien';
ELSIF Moyenne< 17 THEN
        Resultat := 'Très Bien';
ELSE
        Resultat := 'Excelent'
END IF;
```

Structures alternative

- La condition peut comporter les opérateurs suivants:
  - =, < ,>, !=, <=, >=, <>
  - IS [NOT] NULL
  - [NOT] BETWEEN…… AND,
  - [NOT] IN
  - [NOT] LIKE
  - AND, OR, NOT

Exemple:

```
if var1>10 then
        var2:= var1+20;
elseif  var1  between 7 and 8 then
        var2:= 2* var1;
else
        var2:=var1*var1;
end if;
```


Itérations

- **LOOP – EXIT WHEN – END LOOP**

```
LOOP
        …..
        EXIT WHEN CONDITION ;  -- IF (CONDITION) EXIT; END IF
        …..
END LOOP;
```

Exemple: Afficher les entiers de 1 à 10.

```
i:= 1 ;
LOOP
        ….
        i:= i +1 ;
        EXIT WHEN  i >10 ;
        ….
END LOOP;
```

↔ **if i>10 then EXIT; end if;**

Itérations

- **WHILE – LOOP – END**

WHILE *condition* **LOOP**

…..

END LOOP;

Exemple:

WHILE total<= 10000 **LOOP**

……..

SELECT salaire INTO S FROM employe WHERE …

….

total:= total + S ;

END LOOP;

Itérations

- **FOR-IN-LOOP-END**

FOR *compteur* **IN** [**REVERSE**] inf .. sup **LOOP**

…..

END LOOP;

compteur ne peut pas être modifié dans la boucle.

Exemple:

```
FOR I IN 1..10 LOOP
   J:= J* 3 ;
   INSERT INTO T VALUES ( I, J );
END LOOP;
```

```
FOR I IN REVERSE 10..1 LOOP
   J:= J* 3 ;
   INSERT INTO T VALUES ( I, J );
END LOOP;
```

NULL

- Signifie "ne rien faire " , passer à l'instruction suivante

Exemple:

```
IF I>=10 THEN
        NULL
ELSE
        INSERT  INTO  Dept  VALUES ( 50, 'Marketing', 'Casa');
END IF ;
```


Exercices

Pour chacun des exercices suivants, écrire un bloc PL/SQL.

N.B: Si les tables Emp et Dept n'existe pas, exécutez les deux script Schema_DDL.sql et Schema.DML.sql

1. Afficher les entiers de 1 à 10
2. Lire un entier **N** et afficher ses diviseurs.

   Indication: Pour avoir le reste de la division entière, on utilise la fonction **MOD** comme suit: **MOD(***valeur***,** *diviseur***).**
3. Afficher le nombre d'employés de la table Emp.
4. Afficher la moyenne des salaires des employés dont la fonction est saisie au clavier (MANAGER, SALEMAN, ANALYST, …)
5. Afficher le nombre d'employés ayant un salaire en dessous de la moyenne.

Exercice

- Créer les deux tables:

  Livres (Cote number, Titre varchar2(30), Matiere varchar2(30))

  Statistiques (Matiere varchar2(30), Nbre number)
- Insérer les lignes suivantes dans la table Statistiques

  'Informatique', 0

  'Mathématique', 0

  'Physique' , 0
- Insérer les lignes suivantes dans la table Livres:

  1, 'Algorithmique', 'Informatique'

  2, 'SQL', 'Informatique'

  3, 'Algèbre', 'Mathématique'

  4, 'Réseaux Locaux', 'Informatique'

  5, 'Physique Quantique', 'Physique'

Exercice *(suite)*

- Créer un bloc PL/SQL permettant de calculer et afficher le nombre de livres d'informatique dans la table **Livres**.
- Modifier le bloc PL/SQL précédent pour:
  - Calculer le nombre N de livre d'informatique dans la table Livres.
  - Modifier le nombre de livres d'informatique dans la table Statistiques.
- Créer un bloc PL/SQL permettant de:
  - Calculer le nombre N de livre de physique dans la table Livres.
  - Modifier la ligne de la table 'Statistiques' dont la matière est 'physique' pour que la valeur de la colonne Nbre soit N