PL / SQL
Les blocs anonymes

Ines BAKLOUTI

ines.baklouti@esprit.tn

Ecole Supérieure Privée d'Ingénierie et de Technologies

Plan

Introduction

Rappel: le language SQL

SQL: Structured Query Language

- LID: un langage d'interrogation de la base (ordre SELECT)
- LDD: un langage de définition des données (ordres CREATE, ALTER, DROP),
- LMD: un langage de manipulation des données (ordres INSERT, UPDATE, DELETE).

Définition: PL/SQL

Le langage PL/SQL est une extension procédurale du langage SQL (propre à oracle).

Plan

Modèles de programmes en PL/SQL

- Parmi les modèles de programme PL/SQL on trouve:
  - Les blocs anonymes,
  - Les sous programmes (procédures et fonctions),
  - Les procédures et les fonctions stockées,
  - Les triggers,
  - Les packages,
  - ...
- Les programmes écrits en PL/SQL respectent tous une structure en blocs prédéterminés.

Plan

Structure d'un bloc PL/SQL

DECLARATION

DECLARE - Optionnelle

```
/* déclaration variables, constantes, types,
curseurs,... */
```

EXECUTABLE

BEGIN - Obligatoire

```
/* exécution des traitements */
.....
```

EXCEPTION - Optionnelle

```
/* traitement des exception */
```

END - Obligatoire;

```
/
```

Structure d'un bloc PL/SQL

Exemple

```
DECLARE
v_last_name varchar(20);
BEGIN
Select last_name into v_last_name from employees where
employee_id=124;
END ;
/
```

- Chaque instruction SQL ou PL/SQL doit se terminer par un point virgule (;)
- Le slash (/) pour exécuter un bloc PL/SQL anonyme

La section déclaration

- Débute par le mot **DECLARE** et permet la déclaration des variables et constantes.

Exemple

No_Produit Number ;

Désignation VARCHAR2(20) ;

PU NUMBER(6,2) := 100.00;

Les variables PL/SQL

- INTEGER – max 38chiffres
- NUMBER – max 125 chiffres
- BINARY_INTEGER : -2147483647 - +2147483647
- NATURAL :0 - 2147483647
- POSITIVE : 1 - 2147483647
- CHAR; – Max 32767 caractères
- VARCHAR2; – Max 32767 caractères
- LONG; – Max 2147483647 caractères
- DATE; – 4712 avant AVJC à 9999 APJC
- BOOLEAN :TRUE, FALSE, NULL, NOT NULL
- %TYPE : type de variable équivalent au type de colonne d'une table ou d'une autre variable
- %ROWTYPE : type de variable équivalent à une ligne d'une table

Exemples

```
DECLARE
    v_remise CONSTANT real := 0.10;
    v_hiredate date;
    g_deptno number(2) NOT NULL := 10;
    v_integer integer; – max 38chiffres
    v_number1 number; – max 125 chiffres
    v_number2 number(38,3);
    v_bool boolean; – valeurs possibles TRUE, FALSE, NULL et NOT NULL
    v_char char(5); – Max 32767 caractères
    v_varchar2 varchar2(20); – Max 32767 caractères
    v_long LONG; – Max 2147483647 caractères
    v_date date; – Les dates peuvent aller de -4712 avant AVJC à 9999 APJC
    v_last_name employees.last_name%type;
    v_employee employees%rowtype
    v_n1 number(5,3);
    v_n2 v_n1%type;
```

Le type RECORD

- Les enregistrements PL/SQL sont des types composites définis par l'utilisateur. Pour les utiliser il faut:
  - déclarer l'enregistrement dans la section déclarative d'un bloc PL/SQL en utilisant le mot clé TYPE
  - déclarer et éventuellement initialiser les composants internes de ce type d'enregistrement

Déclaration d'un type record

```
TYPE record_name IS RECORD

(field_1 [, field_2, ... field_N]...);


new_record    record_name;
```

Exemple

```
TYPE emp_record_type IS RECORD
(last_name VARCHAR2(25),
job_id VARCHAR2(10),

salary employees.salary%type);


emp_record emp_record_type;
```

Règles de nommination des variables

- Deux variables peuvent porter le même nom si elles sont dans des blocs distincts
- Les noms des variables doivent être différents des noms des colonnes et des tables utilisés dans le bloc
- L'identifiant ne doit pas dépasser 30 caractères
- Le premier caractère doit être une lettre, les autres peuvent être des lettres, des nombres ou des caractères spéciaux.

Déclaration et initialisation des variables

- Les variables sont déclarées et initialisées dans la section déclarative
- Initialisation à l'aide de l'opérateur d'affectation (:=) ou du mot réservé DEFAULT
- Déclarer un seul identifiant par ligne
- Déclarer une constante en utilisant le mot clé CONSTANT
- Initialisation des variables si on a:
  - Mot clé DEFAULT
  - Contrainte NOT NULL
  - Constante

Exemples

```
DECLARE
v_integer integer := 12345;
v_bool Boolean :=TRUE;
v_char varchar(30) NOT NULL := 'SGBD';
v_date date DEFAULT '01-Janv.-2009';
V_constant CONSTANT:=10;
```

Affectation de valeurs aux variables

- De nouvelles valeurs sont affectées aux variables dans la section exécutable (BEGIN .. END)

Exemples

```
BEGIN
v_integer := 12345;
v_number1 := 1234567.453;
v_number2 := 1.9999E+15;
v_bool :=TRUE;
v_char := 'SGBD';
v_long := 'Cours PL/SQL';
v_date := '21-févr.-2008';
END;
```

La portée des variables

```
DECLARE
v1 integer := 100;
v2 varchar(5) :='A';
BEGIN
      DECLARE
            v1 integer;
            v3 varchar2(10):=v2||'1';
            v4 varchar2(10):='B';
      BEGIN
            dbms_output.put_line('v1(1) =' || v1 || chr(10) ||
            'v2(1) =' || v2 || chr(10) || 'v3(1) =' || v3);
      END;
dbms_output.put_line('v1(2) =' || v1 || chr(10) || 'v2(2) =' || v2);
dbms_output.put_line('v3(2) =' || v3);
END;
/
```

La section exécutable

- Intègre différents types d'instructions:
  - Instruction d'affectation
  - Instruction de contrôle de flux (boucles, structures de contrôle)
  - Instruction SQL
  - Instruction curseur

Les structures de contrôle

IF .. THEN .. END IF

IF *condition* **THEN**
– *Instructions*
END IF;

IF .. THEN .. ELSE .. END IF

IF *condition* **THEN**
– *Instructions*
ELSE
– *Instructions*
END IF;

IF .. THEN .. ELSEIF .. THEN .. ELSE .. END IF

IF *condition* **THEN**
– *Instructions*
ELSIF *condition2* **THEN**
– *Instructions*
ELSE
– *Instructions*
END IF;

Les structures de contrôle

Exemple 1

```
DECLARE
v1 integer := 1100;
v2 integer := 200;
BEGIN
IF v1 < v2 THEN
dbms_output.put_line('v1 < v2');
ELSE
dbms_output.put_line('v2 <= v1');
END IF;
END;
/
```

Les structures de contrôle

Exemple 2

```
DECLARE
v1 number:= 685;
v2 number := 125;
v3 number :=870;
BEGIN
IF v1 < v2 THEN
        IF v2<v3 then
                dbms_output.put_line('v1 < v2 < v3');
        ELSIF v3 < v1 then
                dbms_output.put_line('v3 < v1 < v2');
        ELSE
                dbms_output.put_line('v1 <=v3 <v2');
        END IF;
ELSIF v1 < v3 then
        dbms_output.put_line('v2 < v1 < v3');
ELSIF v3<v2 then
        dbms_output.put_line('v3 < v2 <= v1');
ELSE
        dbms_output.put_line('v2 <= v3 <= v1');
END IF;
END;
/
```

Les structures de contrôle

CASE expression

```
CASE expression
WHEN valeur1 THEN
–instructions1;
WHEN valeur2 THEN
–instructions2;
...
ELSE
–instructionsN;
END CASE;
```

CASE

```
CASE
WHEN expr1 THEN
–instructions1;
WHEN expr2 THEN
–instructions2;
...
ELSE
–instructionsN;
END CASE;
```

Les structures de contrôle

Exemple 1

```
DECLARE
v1 integer := 4;
BEGIN
CASE v1
WHEN 1 THEN
dbms_output.put_line('A');
WHEN 2 THEN
dbms_output.put_line('B');
WHEN 3 THEN
dbms_output.put_line('C');
ELSE
dbms_output.put_line('X');
END CASE;
END;
/
```

Exemple 2

```
DECLARE
v1 integer := 18;
BEGIN
CASE
WHEN v1<5 THEN
dbms_output.put_line('A');
WHEN v1<10 THEN
dbms_output.put_line('B');
WHEN v1<15 THEN
dbms_output.put_line('C');
ELSE dbms_output.put_line('X');
END CASE;
END;
/
```

Les structures de contrôle

WHILE .. LOOP .. END LOOP

WHILE condition LOOP

–instructions;

END LOOP;

Exemple

```
DECLARE
v1 integer :=1;
BEGIN
WHILE v1 <10 LOOP
dbms_output.put_line(v1);
v1 := v1+1;
END LOOP;
END;
/
```

Les structures de contrôle

LOOP .. EXIT WHEN .. END LOOP

LOOP
–instructions;
EXIT WHEN condition
–instructions;
END LOOP;

Exemple

```
DECLARE
v1 integer :=1;
BEGIN
LOOP
dbms_output.put_line(v1);
EXIT WHEN v1 =10;
v1 := v1+1;
END LOOP;
END;
/
```

Les structures de contrôle

FOR .. IN .. LOOP .. END LOOP

FOR compteur IN [REVERSE] inf .. sup LOOP

–instructions;

END LOOP;

Exemple

```
DECLARE
v1 integer :=1;
BEGIN
FOR v1 IN 1..10 LOOP
dbms_output.put_line(v1);
END LOOP;
END;
/
```

L'instruction SELECT

SELECT col1, col2, .., colN INTO v_col1, v_col2, .., v_colN FROM nom_table1, nom_table2;

- Utilisation de la clause **INTO** pour identifier les variables PL/SQL qui doivent recevoir des valeurs des colonnes des tables d'une base de données
- La requête select doit retourner **une seule ligne**

Exemple

```
DECLARE
salaire_moy employees.salary%type;
BEGIN
Select avg(sal) into salaire_moy From employees where department_id=10;
dbms_output.put_line('Le salaire moyen des employés du département 10 est : ' ||
to_char(salaire_moy ,'99999.99'));
END ;
/
```

L'instruction SELECT

Exemple

```
DECLARE
v_emp employees%ROWTYPE;
BEGIN
Select * into v_emp From employees where employee_id = 124;
dbms_output.put_line('Nom employé :' || v_emp.last_name || chr(10)||
'Fonction : ' || v_emp.job_id || chr(10)|| 'Departement : '||
v_emp.department_id ||chr(10)|| 'Date recrutement : '||
to_char(v_emp.hire_date, 'dd/mm/yyyy') || chr(10) || 'Salaire : ' ||
v_emp.salary );
END;
/
```