

Lucrarea 9: Comenzi procedurale. Tipul Table si Record

Comenzi procedurale in PL/SQL

Limbajul SQL a fost conceput ca neprocedural, axat pe intrebari pentru a obtine informatii din BD folosind operatorii algebrei relationale Select, Project si Join.

In conditiile dezvoltarii limbajelor procedurale pentru accesul la BD (VdBase, Vfox, Delphi, C++ Buidar) s-a impus definirea si implementarea unui limbaj procedural care sa fie o extensie a SQL. Acesta este PL/SQL care are la baza limbajul ADA numit DIANA - Descriptive Intermediate Attributed Notation for ADA.

PL/SQL este intradevar un limbaj intermediar:

- Accepta comenzile SQL care le trimite spre Oracle SQL Server pentru executie, fiindca presupun controlul drepturilor de acces la obiectele BD si utilizarea informatiilor din dictionarul BD
- Accepta variabile, comenzi procedurale, proceduri, structuri de date, accesul la inregistrările din tabele prin cursoare, utilizarea obiectelor utilizator

Comenzile procedurale de baza sunt IF si LOOP care permit realizarea unor secvente de program ramificate si ciclice. Secventele de program pot contine comenzi SQL, functii satandard PL/SQL, calcule complexe folosind constate, variabile si expresii. PL/SQL executa toate comezile proprii, exceptand comenzile SQL care le trimite spre serverul SQL si receptioneaza rezultatele.

Forma generala acceptata de PL/SQL pentru Select este

```
SELECT lista_campuri INTO lista_variabile FROM tabela ,...WHERE cond;
```

Conditia trebuie sa fie suficient de restrictiva pentru a returna un singur rand.

In caz contrar da eroare: Too Many Rows!

Daca rezulta mai multe randuri de tabela rezultat se va folosi CURSOR (Lucrarea 10), sau se va folosi o tabela pentru rezultate in care se va scrie sub forma:

```
INSERT INTO tab_rez SELECT lista_campuri FROM lista_tabele WHERE cond;
```

Aceasta comanda este realizata complet de SQL Server care nu returneaza nimic.

Comanda IF are 2 forme:

Prima forma este cea clasica

```
IF conditie THEN      -- Daca conditia este indeplinita se executa secventa
...   secventa
...
[ELSE]                -- Daca conditia nu este indeplinita
....   Secventa_else  -- nu se executa nimic sau se executa secventa_else
....
END IF;
```

A 2-a forma este o varianta de Do Case care evita insiruirea de IF-uri:

```

IF conditie1 THEN           -- Se executa secventa i  daca este
...   secventa1                -- indeplinita conditia i
...
ELSIF conditie2
...   secventa2
...
ELSIF conditie3
...   secventa3
...
ELSIF conditie4
...   secventa4
...

[ELSE]                        -- Daca nici o conditie nu este indeplinita
....   Secventa_else          -- nu se executa nimic sau se executa secventa_else
....
END IF;

```

Comanda LOOP are 3 forme de baza

- Cu numar fix de cicluri:

```

FOR k IN [REVERSE] k1..k2 LOOP
....   secventa de comenzi
....
END LOOP;

```

- Cu verificarea conditiei de iesire din ciclu la inceput

```

WHILE conditie LOOP
....   secventa de comenzi
....
END LOOP;

```

- Cu verificarea conditiei de iesire din ciclu la sfarsit

```

LOOP
....   secventa de comenzi
....
EXIT [WHEN conditie];
END LOOP;

```

Exemple de utilizare LOOP:

*** Program LOOP_FOR**

* Calculul sumei numerelor din intervalul k1,k2 folosind FOR ...LOOP

SET SERVEROUTPUT ON

DECLARE

K INTEGER;

K1 INTEGER := 1;

K2 INTEGER :=20;

S NUMBER(7) :=0;

BEGIN

S:=0;

FOR k IN k1..k2 LOOP

S:=S+k;

END LOOP;

DBMS_OUTPUT.PUT_LINE('Suma= ' || S || ' pentru intervalul ' || k1 || ' ' || k2);

END;

/

*** Program LOOP_WHILE**

* Calculul sumei numerelor din intervalul k1,k2 folosind WHILE conditie ...LOOP

SET SERVEROUTPUT ON

DECLARE

K INTEGER;

K1 INTEGER := 1;

K2 INTEGER :=20; -- :=&k2

S NUMBER(7) :=0;

BEGIN

S:=0;

k:=k1;

WHILE k<=k2 LOOP

S:=S+k;

k:=k+1;

END LOOP;

DBMS_OUTPUT.PUT_LINE('Suma= ' || S || ' pentru intervalul ' || k1 || ' ' || k2);

END;

/

*** Program LOOP_EXIT**

* Suma numerelor din intervalul k1,k2 folosind LOOP ... EXIT WHEN conditie

SET SERVEROUTPUT ON

DECLARE

K INTEGER;

K1 INTEGER := 1;

K2 INTEGER :=20;

S NUMBER(7) :=0;

BEGIN

S:=0;

k:=k1;

LOOP

S:=S+k;

k:=k+1;

EXIT WHEN k>k2;

```

END LOOP;
DBMS_OUTPUT.PUT_LINE( 'Suma= ' || S || ' pentru intervalul ' || k1 || ' ' || k2);
END;
/

```

Definire si utilizare tip tabela indexata

In PL/SQL se pot declara tabele indexate care nu au toate elementele completate.
Se memoreaza numai pozitiiile care au valori ne nule.

```

Var%TYPE
TYPE tabela IS TABLE OF tip_data INDEX BY BINARY_INTEGER;
Col%TYPE
Tab.col%TYPE

```

```

SET SERVEROUTPUT ON
DECLARE
TYPE Tab IS TABLE OF integer INDEX BY BINARY_INTEGER;
T1 tab; -- declarare T1 ca tip tab definit anterior
K INTEGER;
K1 INTEGER := 1;
K2 INTEGER :=20;
S NUMBER(7) :=0;
BEGIN
FOR k IN k1..k2 LOOP -- completare tabela
T1(k):=k;
END LOOP;
S:=0;
FOR k IN k1..k2 LOOP -- Suma elemente tabela
S:=S+T1(k);
END LOOP;
DBMS_OUTPUT.PUT_LINE( 'Suma= ' || S || ' pentru intervalul ' || k1 || ' ' || k2);
END;
/
Suma= 210 pentru intervalul 1 20

```

* Definire tabela de nume de salariati:

```

DECLARE
TYPE Tsal IS TABLE OF EMP.Ename%Type INDEX BY BINARY_INTEGER;
Sal1 Tsal;
BEGIN
Sal1(734):='VASILE';
Sal1(562):='Dumitru';
DBMS_OUTPUT.PUT_LINE( Sal1(734)|| ' ' || Sal1(562));
END;
/
Tipul RECORD

```

Un record este o variabila compusa formata din campuri de tipuri diferite.
Se defineste initial un tip record si apoi se declara variabile de tipul definit.

```

TYPE tip_record IS RECORD
    (cimp1 tip1,cimp2 tip2,.....);
R1 tip_record;           -- definire variabila record

```

Se poate declara o variabila record de tipul randurilor unei tabele.

```

R2    Emp%RowType;

```

Referirea la un camp dintr-o variabila Record se face sub forma:

Nume_record.Camp de exemplu: R2.Ename

*** Program Rec1 *** Se utilizeaza variabile record declarate pentru citire din fisier

```

SET SERVEROUTPUT ON

```

```

DECLARE

```

```

/* declarare tip record pentru date personale*/

```

```

TYPE R_pers IS RECORD

```

```

    (NUME      Varchar2(20),
     Functia    Varchar2(10),
     Salar      Number(7));

```

```

R1    R_pers;           -- Declarare record de tipul R_pers

```

```

R2    Emp%RowType;      -- Declarare record salariat

```

```

-- Declarare tip tabela de record-uri salariati

```

```

TYPE T_sal IS TABLE OF Emp%Rowtype Index By Binary_Integer;

```

```

Tpers T_sal;

```

```

BEGIN

```

```

-- Citire inregistrare in recordul R1 respectiv R2

```

```

    SELECT Ename,Job,Sal INTO R1 FROM Emp WHERE Empno=7900;

```

```

    SELECT * INTO R2 FROM EMP WHERE Empno=7902;

```

```

-- Afisare date din record R1

```

```

DBMS_OUTPUT.PUT_LINE( R1.Nume || ' Functia: '||R1.Functia||
    ' Salar: '||R1.Salar);

```

```

DBMS_OUTPUT.PUT_LINE( 'Nume: ' ||R2.Ename||'Marca: ' ||R2.Empno|| '
Functia: '||R2.Job|| ' Salar: '||R2.Sal|| ' Data angajarii: '|| R2.Hiredate);

```

```

END;

```

```

/

```

JAMES Functia: CLERK Salar: 950

Nume: FORDMarca: 7902 Functia: ANALYST Salar: 3000

Data angajarii: 03-DEC-81

PL/SQL procedure successfully completed.