

Controlul execuției unui bloc *PL/SQL*

PL/SQL este un limbaj cu structură de bloc, adică programele sunt compuse din blocuri care pot fi complet separate sau imbricate. Structura unui bloc poate fi obținută combinând subprograme, pachete, blocuri imbricate. Blocurile pot fi folosite în utilitățile *Oracle*.

Pentru modularizarea unui program este necesară:

- gruparea logică a instrucțiunilor în blocuri;
- imbricarea de subblocuri în blocuri mai mari;
- descompunerea unei probleme complexe într-o mulțime de module logice și implementarea acestora cu ajutorul blocurilor;
- plasarea în biblioteci a codului *PL/SQL* reutilizabil, de unde poate fi folosit de aplicații;
- depunerea codului într-un *server Oracle*, de unde este accesibil oricărei aplicații care interacționează cu baza de date *Oracle*.

Un program *PL/SQL* poate cuprinde unul sau mai multe blocuri. Un bloc poate fi anonim sau neanonim.

Blocurile **anonime** sunt blocuri *PL/SQL* fără nume, care sunt construite dinamic și sunt executate o singură dată. Acest tip de bloc nu are argumente și nu returnează un rezultat. Ele sunt declarate într-un punct al aplicației, unde vor fi executate (trimise motorului *PL/SQL*). În blocurile anonime pot fi declarate proceduri și funcții *PL/SQL*.

Blocurile anonime pot să apară într-un program ce lucrează cu precompilator sau în *SQL*Plus*. De obicei, blocurile anonime sunt plasate într-un fișier, iar apoi fișierul este executat din *SQL*Plus*. De asemenea, declanșatorii din componentele *Developer Suite* constau din astfel de blocuri.

Blocurile **neanonime** sunt fie blocuri cu nume (etichetate) construite static sau dinamic și executate o singură dată, fie subprograme, pachete sau declanșatori.

Subprogramele sunt proceduri sau funcții depuse în baza de date. Aceste blocuri sunt executate de mai multe ori și, în general, nu mai sunt modificate după ce au fost construite. Procedurile și funcțiile stocate sunt depuse pe *server-ul Oracle*, acceptă parametri și pot fi apelate prin nume. Procedurile și funcțiile aplicație sunt depuse într-o aplicație *Developer Suite* sau într-o bibliotecă.

Pachetele (stocate sau aplicație) sunt blocuri neanonime care grupează proceduri, funcții, cursoare, tipuri, constante, variabile într-o unitate logică, în baza de date.

Declanșatorii sunt blocuri *PL/SQL* neanonime depuse în baza de date, care pot fi asociați bazei, iar în acest caz sunt executați implicit ori de câte ori apare un anumit eveniment declanșator (de exemplu, instrucțiuni *INSERT*, *UPDATE* sau *DELETE* ce se execută asupra unui tabel al bazei de date) sau pot fi asociați unei aplicații (de exemplu, declanșator *SQL*Forms*), ceea ce presupune că se execută automat, în funcție de anumite condiții sistem.

Structura unui bloc *PL/SQL*

Un bloc *PL/SQL* este compus din trei secțiuni distincte.

- Secțiunea declarativă (opțională) conține declarații pentru toate variabilele, constantele, cursoarele și erorile definite de utilizator la care se face referință în secțiunea executabilă sau chiar în cea declarativă. De asemenea, pot fi declarate subprograme locale care sunt vizibile doar în blocul respectiv.
- Secțiunea executabilă conține instrucțiuni neprocedurale *SQL* pentru prelucrarea datelor din baza de date și instrucțiuni *PL/SQL* pentru prelucrarea datelor în cadrul blocului.
- Secțiunea pentru tratarea erorilor (opțională) specifică acțiunile ce vor fi efectuate atunci când în execuția blocului apar erori sau condiții anormale.

Blocul *PL/SQL* are următoarea structură generală:

```
[<<nume_bloc>>]
[DECLARE
    instrucțiuni de declarare]
BEGIN
    instrucțiuni executabile (SQL sau PL/SQL)
[EXCEPTION
    tratarea erorilor]
END [nume_bloc];
```

Dacă blocul *PL/SQL* este executat fără erori, invariant va apărea mesajul:

PL/SQL procedure successfully completed

Compatibilitate *SQL*

Din punct de vedere al compatibilității dintre *PL/SQL* și *SQL*, se remarcă următoarele reguli de bază:

- *PL/SQL* furnizează toate comenzile *LMD* ale lui *SQL*, comanda *SELECT* cu clauza *INTO*, comenzile *LCD*, funcțiile, pseudocoloanele și operatorii *SQL*;
- *PL/SQL* nu furnizează comenzile *LDD*.

Totuși, în ultimele sale versiuni, *Oracle* permite folosirea dinamică a comenzilor *SQL*, utilizând tehnica oferită de *SQL* dinamic. În felul acesta, orice comandă *SQL* (inclusiv comandă *LDD*) poate să fie utilizată în *PL/SQL*.

Majoritatea funcțiilor *SQL* sunt disponibile în *PL/SQL*. Există însă funcții specifice *PL/SQL*, cum sunt funcțiile *SQLCODE* și *SQLERRM*. De asemenea, există funcții *SQL* care nu sunt disponibile în instrucțiuni procedurale (*DECODE*, funcțiile grup), dar care sunt disponibile în instrucțiunile *SQL* dintr-un bloc *PL/SQL*. *SQL* nu poate folosi funcții sau atribute specifice *PL/SQL*.

Funcțiile grup trebuie folosite cu atenție, deoarece clauza *GROUP BY* nu are sens să apară în instrucțiunea *SELECT ... INTO*. *Oracle9i* introduce clauza *OVER*, care permite ca funcția grup căreia îi este asociată să fie considerată o funcție analitică (poate returna mai multe linii pentru fiecare grup).

Următoarele funcții *SQL* nu sunt permise în *PL/SQL*: *WIDTH_BUCKET*, *BIN_TO_NUM*, *COMPOSE*, *DECOMPOSE*, *TO_LOB*, *DECODE*, *DUMP*, *EXISTSNode*, *TREAT*, *NULLIF*, *SYS_CONNECT_BY_PATH*, *SYS_DBURIGEN*, *EXTRACT*.

Instrucțiuni *PL/SQL*

Orice program poate fi scris utilizând structuri de control de bază care sunt combinate în diferite moduri pentru rezolvarea problemei propuse. *PL/SQL* dispune de comenzi ce permit controlul execuției unui bloc. Instrucțiunile limbajului pot fi: iterative (*LOOP*, *WHILE*, *FOR*), de atribuire (*:=*), condiționale (*IF*, *CASE*), de salt (*GOTO*, *EXIT*) și instrucțiunea vidă (*NULL*).

Observații:

- Comentariile sunt ignorate de compilatorul *PL/SQL*. Există comentarii pe o singură linie, prefixate de simbolurile „--“, care încep în orice punct al liniei și se termină la sfârșitul acesteia. De asemenea, există comentarii pe mai multe linii, care sunt delimitate de simbolurile „/*“ și „*/“. Nu se admit comentarii imbricate.
- Caracterul „;“ este separator pentru instrucțiuni.
- Atât operatorii din *PL/SQL*, cât și ordinea de execuție a acestora, sunt identici cu cei din *SQL*. În *PL/SQL* este introdus un nou operator („**“) pentru ridicare la putere.
- Un identificator este vizibil în blocul în care este declarat și în toate subblocurile, procedurile și funcțiile imbricate în acesta. Dacă blocul nu găsește identificatorul declarat local, atunci îl caută în secțiunea declarativă a blocurilor care includ blocul respectiv și niciodată nu caută în blocurile încuibărite în acesta.
- Comenzile *SQL*Plus* nu pot să apară într-un bloc *PL/SQL*.

- În comanda *SELECT* trebuie specificate variabilele care recuperează rezultatul acțiunii acestei comenzi. În clauza *INTO*, care este obligatorie, pot fi folosite variabile *PL/SQL* sau variabile de legătură.
- Referirea la o variabilă de legătură se face prin prefixarea acesteia cu simbolul „:”.
- Cererea dintr-o comandă *SELECT* trebuie să returneze o singură linie drept rezultat. Atunci când comanda *SELECT* întoarce mai multe linii, apare eroarea *TOO_MANY_ROWS*, iar în cazul în care comanda nu găsește date se generează eroarea *NO_DATA_FOUND*.
- Un bloc *PL/SQL* nu este o unitate tranzacțională. Într-un bloc pot fi mai multe tranzații sau blocul poate face parte dintr-o tranzație. Acțiunile *COMMIT*, *SAVEPOINT* și *ROLLBACK* sunt independente de blocuri, dar instrucțiunile asociate acestor acțiuni pot fi folosite într-un bloc.
- *PL/SQL* nu suportă comenzile *GRANT* și *REVOKE*, utilizarea lor fiind posibilă doar prin *SQL* dinamic.

Fluxul secvențial de execuție a comenzilor unui program *PL/SQL* poate fi modificat cu ajutorul structurilor de control: *IF*, *CASE*, *LOOP*, *FOR*, *WHILE*, *GOTO*, *EXIT*.

Instrucțiunea de atribuire

Instrucțiunea de atribuire se realizează cu ajutorul operatorului de asignare (*:=*) și are forma generală clasică (*variabila := expresie*). Comanda respectă proprietățile instrucțiunii de atribuire din clasa *LG3*. De remarcat că nu poate fi asignată valoarea *null* unei variabile care a fost declarată *NOT NULL*.

Exemplu:

Următorul exemplu prezintă modul în care acționează instrucțiunea de atribuire în cazul unor tipuri de date particulare.

```
DECLARE
    alfa INTERVAL YEAR TO MONTH;
BEGIN
    alfa := INTERVAL '200-7' YEAR TO MONTH;
    -- alfa ia valoarea 200 de ani si 7 luni
    alfa := INTERVAL '200' YEAR;
    -- pot fi specificati numai anii
    alfa := INTERVAL '7' MONTH;
    -- pot fi specificate numai lunile
    alfa := '200-7';
    -- conversie implicita din caracter
END;
```

```

DECLARE
    beta  opera%ROWTYPE;
    gama  opera%ROWTYPE;
    cursor epsilon IS SELECT * FROM opera;
    delta  epsilon%ROWTYPE;
BEGIN
    beta := gama;  -- corect
    gama := delta; -- incorect???-testati!
END;
```

Instrucțiunea *IF*

Un program *PL/SQL* poate executa diferite porțiuni de cod, în funcție de rezultatul unui test (predicat). Instrucțiunile care realizează acest lucru sunt cele condiționale (*IF*, *CASE*).

Structura instrucțiunii *IF* în *PL/SQL* este similară instrucțiunii *IF* din alte limbaje procedurale, permițând efectuarea unor acțiuni în mod selectiv, în funcție de anumite condiții. Instrucțiunea *IF-THEN-ELSIF* are următoarea formă sintactică:

```

IF condiție1 THEN
    secvența_de_comenzi_1
[ELSIF condiție2 THEN
    secvența_de_comenzi_2]
...
[ELSE
    secvența_de_comenzi_n]
END IF;
```

O secvență de comenzi din *IF* este executată numai în cazul în care condiția asociată este *TRUE*. Atunci când condiția este *FALSE* sau *NULL*, secvența nu este executată. Dacă pe ramura *THEN* se dorește verificarea unei alternative, se folosește ramura *ELSIF* (atenție, nu *ELSEIF*) cu o nouă condiție. Este permis un număr arbitrar de opțiuni *ELSIF*, dar poate apărea cel mult o clauză *ELSE*. Aceasta se referă la ultimul *ELSIF*.

Exemplu:

Să se specifice dacă o galerie este *mare*, *medie* sau *mica* după cum numărul operelor de artă expuse în galeria respectivă este mai mare decât 200, cuprins între 100 și 200 sau mai mic decât 100.

```

SET SERVEROUTPUT ON
DEFINE p_cod_gal = 753
DECLARE
    v_cod_galerie    opera.cod_galerie%TYPE := &p_cod_gal;
    v_numar          NUMBER(3) := 0;
    v_comentariu     VARCHAR2(10);
BEGIN
    SELECT COUNT(*)
    INTO    v_numar
    FROM    opera
    WHERE   cod_galerie = v_cod_galerie;
    IF v_numar < 100 THEN
        v_comentariu := 'mica';
    ELSIF v_numar BETWEEN 100 AND 200 THEN
        v_comentariu := 'medie';
    ELSE
        v_comentariu := 'mare';
    END IF;
    DBMS_OUTPUT.PUT_LINE('Galeria avand codul ' ||
        v_cod_galerie || ' este de tip ' || v_comentariu);
END;
/
SET SERVEROUTPUT OFF

```

Instrucțiunea CASE

Oracle9i furnizează o nouă comandă (*CASE*) care permite implementarea unor condiții multiple. Instrucțiunea are următoarea formă sintactică:

```

[<<eticheta>>]
CASE test_var
    WHEN valoare_1 THEN secvența_de_comenzi_1;
    WHEN valoare_2 THEN secvența_de_comenzi_2;
    ...
    WHEN valoare_k THEN secvența_de_comenzi_k;
    [ELSE altă_secvență;]
END CASE [eticheta];

```

Se va executa *secvența_de_comenzi_p*, dacă valoarea selectorului *test_var* este *valoare_p*. După ce este executată secvența de comenzi, controlul va trece la următoarea instrucțiune după *CASE*. Selectorul *test_var* poate fi o variabilă sau o expresie complexă care poate conține chiar și apeluri de funcții.

Clauza *ELSE* este opțională. Dacă această clauză este necesară în implementarea unei probleme, dar totuși lipsește, iar *test_var* nu ia nici una dintre valorile ce apar în clauzele *WHEN*, atunci se declanșează eroarea predefinită *CASE_NOT_FOUND (ORA - 06592)*.

Comanda *CASE* poate fi etichetată și, în acest caz, eticheta poate să apară la sfârșitul clauzei *END CASE*. De remarcat că eticheta după *END CASE* este permisă numai în cazul în care comanda *CASE* este etichetată.

Selectorul *test_var* poate să lipsească din structura comenzii *CASE*, care în acest caz va avea următoarea formă sintactică:

```
[<<eticheta>>]
```

CASE

WHEN *condiție_1* **THEN** *secvența_de_comenzi_1*;

WHEN *condiție_2* **THEN** *secvența_de_comenzi_2*;

...

WHEN *condiție_k* **THEN** *secvența_de_comenzi_k*;

[**ELSE** *altă_secvență*];

END CASE [*eticheta*];

Fiecare clauză *WHEN* conține o expresie booleană. Dacă valoarea lui *condiție_p* este *TRUE*, atunci este executată *secvența_de_comenzi_p*.

Exemplu:

În funcție de o valoare introdusă de utilizator, care reprezintă abrevierea zilelor unei săptămâni, să se afișeze (în cele două variante) un mesaj prin care este specificată ziua săptămânii corespunzătoare abrevierii respective.

Varianta 1:

```
SET SERVEROUTPUT ON
DEFINE p_zi = x
DECLARE
    v_zi CHAR(2) := UPPER('&p_zi');
BEGIN
    CASE v_zi
        WHEN 'L' THEN DBMS_OUTPUT.PUT_LINE('Luni');
        WHEN 'M' THEN DBMS_OUTPUT.PUT_LINE('Marti');
        WHEN 'MI' THEN DBMS_OUTPUT.PUT_LINE('Miercuri');
        WHEN 'J' THEN DBMS_OUTPUT.PUT_LINE('Joi');
        WHEN 'V' THEN DBMS_OUTPUT.PUT_LINE('Vineri');
        WHEN 'S' THEN DBMS_OUTPUT.PUT_LINE('Sambata');
        WHEN 'D' THEN DBMS_OUTPUT.PUT_LINE('Duminica');
        ELSE DBMS_OUTPUT.PUT_LINE('este o eroare!');
    END CASE;
END;
/
SET SERVEROUTPUT OFF
```

Varianta 2:

```

SET SERVEROUTPUT ON
DEFINE p_zi = x
DECLARE
    v_zi CHAR(2) := UPPER('&p_zi');
BEGIN
    CASE
        WHEN v_zi = 'L' THEN
            DBMS_OUTPUT.PUT_LINE('Luni');
        WHEN v_zi = 'M' THEN
            DBMS_OUTPUT.PUT_LINE('Marti');
        WHEN v_zi = 'MI' THEN
            DBMS_OUTPUT.PUT_LINE('Miercuri');
        WHEN v_zi = 'J' THEN
            DBMS_OUTPUT.PUT_LINE('Joi');
        WHEN v_zi = 'V' THEN
            DBMS_OUTPUT.PUT_LINE('Vineri');
        WHEN v_zi = 'S' THEN
            DBMS_OUTPUT.PUT_LINE('Sambata');
        WHEN v_zi = 'D' THEN
            DBMS_OUTPUT.PUT_LINE('Duminica');
        ELSE DBMS_OUTPUT.PUT_LINE('Este o eroare!');
    END CASE;
END;
/
SET SERVEROUTPUT OFF

```

Oracle8i a implementat suportul pentru expresii *CASE* care sunt permise numai în comenzi *SQL*. În *Oracle9i* poate fi utilizată o construcție *CASE* într-o comandă *SQL* a unui bloc *PL/SQL*.

Expresia *CASE* are sintaxa similară comenzii *CASE*, dar clauzele *WHEN* nu se termină prin caracterul „;”, clauza *END* nu include cuvântul cheie *CASE* și nu se fac atribuiri în clauza *WHEN*.

Exemplu:

```

BEGIN
    FOR j IN (SELECT
                CASE valoare
                    WHEN 1000 THEN 1100
                    WHEN 10000 THEN 11000
                    WHEN 100000 THEN 110000
                    ELSE valoare
                END
            FROM opera)
    ...
    END LOOP;
END;

```

Instrucțiuni iterative

Există trei tipuri de comenzi iterative: ciclarea simplă *LOOP*, ciclarea *WHILE* și ciclarea *FOR*.

Acestea permit repetarea (condiționată sau necondiționată) execuției uneia sau mai multor instrucțiuni. Ciclurile pot fi imbricate pe mai multe niveluri. Ele pot fi etichetate, iar ieșirea din ciclu se poate realiza cu ajutorul comenzii *EXIT*.

Se utilizează:

- comanda *LOOP*, dacă instrucțiunile din cadrul ciclului trebuie să se execute cel puțin o dată;
- comanda *WHILE*, în cazul în care condiția trebuie evaluată la începutul fiecărei iterații;
- comanda *FOR*, dacă numărul de iterații este cunoscut.

Instrucțiunea *LOOP* are următoarea formă sintactică:

LOOP

secvența_de_comenzi;

END LOOP;

Ciclarea simplă cuprinde o mulțime de comenzi incluse între cuvintele cheie *LOOP* și *END LOOP*. Aceste comenzi se execută cel puțin o dată. Dacă nu este utilizată comanda *EXIT*, ciclarea poate continua la infinit.

Exemplu:

Se presupune că a fost creată structura tabelului *org_tab*, constând din două coloane: *cod_tab* de tip *INTEGER*, ce conține un contor al înregistrărilor și *text_tab* de tip *VARCHAR2*, ce conține un text asociat fiecărei înregistrări. Să se introducă 70 de înregistrări în tabelul *org_tab*.

```
DECLARE
  v_contor BINARY_INTEGER := 1;
BEGIN
  LOOP
    INSERT INTO org_tab
    VALUES (v_contor, 'indicele ciclului');
    v_contor := v_contor + 1;
    EXIT WHEN v_contor > 70;
  END LOOP; COMMIT;
END;
```

Instrucțiunea repetitivă *WHILE* permite repetarea unei secvențe de instrucțiuni, atâta timp cât o anumită condiție specificată este adevărată.

Comanda *WHILE* are următoarea sintaxă:

```
WHILE condiție LOOP
    secvența_de_comenzi;
END LOOP;
```

Dacă variabilele care apar în condiție nu se schimbă în interiorul ciclului, atunci condiția rămâne adevărată și ciclul nu se termină.

Când condiția este evaluată ca fiind *FALSE* sau *NULL*, atunci secvența de comenzi nu este executată și controlul trece la prima instrucțiune după *END LOOP*.

Exemplu:

```
DECLARE
    v_contor BINARY_INTEGER := 1;
BEGIN
    WHILE v_contor <= 70 LOOP
        INSERT INTO org_tab
        VALUES (v_contor, 'indicele ciclului');
        v_contor := v_contor + 1;
    END LOOP;
END;
```

Instrucțiunea repetitivă *FOR* (ciclare cu pas) permite executarea unei secvențe de instrucțiuni pentru valori ale variabilei *contor* cuprinse între două limite, *lim_inf* și *lim_sup*. Dacă este prezentă opțiunea *REVERSE*, iterația se face (în sens invers) de la *lim_sup* la *lim_inf*.

Comanda *FOR* are sintaxa:

```
FOR contor_ciclu IN [REVERSE] lim_inf..lim_sup LOOP
    secvența_de_comenzi;
END LOOP;
```

Variabila *contor_ciclu* nu trebuie declarată. Ea este neidentificată în afara ciclului și implicit de tip *BINARY_INTEGER*. Pasul are implicit valoarea 1 și nu poate fi modificat. Limitele domeniului pot fi variabile sau expresii, care să poată fi convertite la întreg.

Exemplu:

În structura tabelului *opera* se va introduce un nou câmp (*stea*). Să se creeze un bloc *PL/SQL* care va reactualiza acest câmp, introducând o stelută pentru fiecare 10000\$ din valoarea unei opere de artă al cărei cod este specificat.

```
ALTER TABLE  opera
ADD  stea      VARCHAR2(20);

DEFINE p_cod_opera = 7777
DECLARE
  v_cod_opera  opera.cod_opera%TYPE := &p_cod_opera;
  v_valoare    opera.valoare%TYPE;
  v_stea       opera.stea%TYPE := NULL;
BEGIN
  SELECT  NVL(ROUND(valoare/10000),0)
  INTO    v_valoare
  FROM    opera
  WHERE   cod_opera = v_cod_opera;
  IF v_valoare > 0 THEN
    FOR i IN 1..v_valoare LOOP
      v_stea := v_stea || '*';
    END LOOP;
  END IF;
  UPDATE  opera
  SET     stea = v_stea
  WHERE   cod_opera = v_cod_opera;
  COMMIT;
END;
```

Instrucțiuni de salt

Instrucțiunea *EXIT* permite ieșirea dintr-un ciclu. Ea are o formă necondițională (ieșire fără condiții) și una condițională. Controlul trece fie la prima instrucțiune situată după clauza *END LOOP* corespunzătoare, fie după instrucțiunea *LOOP* având eticheta *nume_eticheta*.

EXIT [*nume_eticheta*] [***WHEN*** *condiție*];

Numele etichetelor urmează aceleași reguli ca cele definite pentru identificatori. Eticheta se plasează înaintea comenzii, fie pe aceeași linie, fie pe o linie separată. În *PL/SQL* etichetele se definesc prin intercalarea numelui etichetei între caracterele „<<” și „>>” (<<*eticheta*>>).

Exemplu:

```

DECLARE
    v_contor      BINARY_INTEGER := 1;
    raspuns       VARCHAR2(10);
    alt_raspuns    VARCHAR2(10);
BEGIN
    ...
    <<exterior>>
    LOOP
        v_contor := v_contor + 1;
        EXIT WHEN v_contor > 70;
        <<interior>>
        LOOP
            ...
            EXIT exterior WHEN raspuns = 'DA';
            -- se parasesc ambele cicluri
            EXIT WHEN alt_raspuns = 'DA';
            -- se parasesc ciclul interior
            ...
        END LOOP interior;
        ...
    END LOOP exterior;
END;
```

Instrucțiunea *GOTO* determină un salt necondiționat la o instrucțiune executabilă sau la începutul unui bloc care are eticheta specificată în comandă. Instrucțiunea are următoarea formă sintactică:

GOTO *nume_eticheta*;

Nu este permis saltul:

- în interiorul unui bloc (subbloc);
- în interiorul unei comenzi *IF*, *CASE* sau *LOOP*;
- de la o clauză a comenzii *CASE*, la altă clauză a aceleiași comenzi;
- de la tratarea unei excepții, în blocul curent;
- în exteriorul unui subprogram.

Instrucțiunea vidă

Instrucțiunea vidă (*NULL*) este folosită pentru o mai bună lizibilitate a programului. *NULL* este instrucțiunea care nu are nici un efect, marcând faptul că nu trebuie întreprinsă nici o acțiune. Nu trebuie confundată instrucțiunea *NULL* cu valoarea *null*!

Uneori instrucțiunea *NULL* este folosită într-o comandă *IF*, indicând faptul că pentru o anumită clauză *ELSIF* nu se execută nici o acțiune.