ACADEMIA DE STUDII ECONOMICE BUCUREŞTI
FACULTATEA DE CIBERNETICĂ, STATISTICĂ ŞI INFORMATICĂ ECONOMICĂ

Admitere 2017

Programul de masterat profesional de specializare

BAZE DE DATE – SUPORT PENTRU AFACERI

Tematica de concurs CSIE3

PROGRAMAREA ÎN LIMBAJUL PL/SQL

BUCUREŞTI
Iulie 2017

Conf.univ.dr. Iuliana BOTHA

Tematica de concurs CSIE3

NR. CRT	TEMATICA		REFERINŢA BIBLIOGRAFICĂ	PAGINAŢIE
1.	Baze de date relaţionale	Modelul relaţional: structura relaţională a datelor, algebra şi calculul relaţional, restricţii de integritate. Exemplificări în Oracle	[1]	pag. 103-121 pag. 129-143
		Realizarea bazelor de date relaţionale: analiza statică, dinamică şi funcţională; proiectarea structurii conceptuale, logice şi fizice; normalizarea datelor	[1]	pag. 144-186 pag. 197-203
2.	Programarea în limbajul SQL	Actualizarea structurii bazei de date: crearea obiectelor, modificarea proprietăţilor şi ştergerea acestora	[3]	pag. 101-122
		Actualizarea datelor: adăugarea de înregistrări, modificarea valorilor, ştergerea înregistrărilor	[3]	pag. 123-126
		Interogarea datelor: condiţionarea datelor, utilizarea joncţiunilor şi a funcţiilor SQL, gruparea datelor, gestiunea subcererilor	[3]	pag. 127-172
3.	Programarea în limbajul PL/SQL	Elemente de programare procedurală	[2]	pag. 9-52
		Mecanismul de cursor	[2]	pag. 53-77
		Gestiunea subprogramelor: proceduri şi funcţii	[2]	pag. 103-118

Bibliografie

NR. CRT	REFERINŢA BIBLIOGRAFICĂ	
[1]	Lungu I., Bâra A., Bodea C., Botha I., Diaconiţa V., Florea A., Velicanu A.	Tratat de baze de date. Vol I. Baze de date. Organizare, proiectare şi implementare, Editura ASE, Bucureşti, 2011, ISBN 978-606-505-472-1, ISBN volum 978-606-505-481-3
[2]	Bâra A., Botha I., Diaconiţa V., Lungu I., Velicanu A.	Baze de date. Limbajul PL/SQL, Editura ASE, Bucureşti, 2009, ISBN 978-606-505-263-5
[3]	Lungu I.	Baze de date Oracle. Limbajul SQL, Editura ASE, Bucureşti, 2005, ISBN 973-594-684-X

Agenda

- Introducere în limbajul PL/SQL
- Elemente de programare procedurală
- Mecanismul de cursor în SGBD Oracle
- Gestiunea subprogramelor în SGBD Oracle
- Exemple de teste grilă *

* *Teste grilă propuse la examenele de admitere din sesiunile anterioare*

Introducere în limbajul PL/SQL

- PL/SQL este un limbaj de programare procedural care extinde limbajul descriptiv SQL şi care conţine următoarele elemente:
  - Blocuri anonime
  - Proceduri
  - Funcţii
  - Pachete care grupează funcţii şi proceduri
  - Declanşatori

Structura generală a unui bloc PL/SQL

DECLARE *--opţional*
--secţiunea declarativă a blocului;
--cuprinde declaraţii de variabile, cursori, excepţii...;
BEGIN *--obligatoriu*
--secţiunea executabilă a blocului;
--cuprinde comenzi descriptive SQL, comenzi procedurale şi structuri de programare PL/SQL;
EXCEPTION *--opţional*
--secţiunea de tratare a excepţiilor;
--cuprinde acţiuni care se execută în momentul apariţiei unei excepţii sau erori;
END; *--obligatoriu*

Introducere în limbajul PL/SQL
Variabile PL/SQL

- declararea variabilelor se realizează în zona declarativă (delimitată prin DECLARE) a blocului;
- iniţializarea se poate face la declarare sau în zona de execuţie (între BEGIN şi END);
- variabilele vor fi vizibile în restul blocului, respectiv şi în blocurile incluse în el, mai puţin în sub-blocurile în care numele lor este redefinit (ca în majoritatea limbajelor de programare structurate, semnificaţia unui nume definit de utilizator într-un bloc/sub-bloc este dată de cea mai apropiată declaraţie anterioară locului folosirii);
- toate variabilele PL/SQL au un tip de dată, restricţii şi un şir valid de valori;
- declararea şi iniţializarea se realizează astfel:

  nume_variabila [CONSTANT] TIP_DATA [NOT NULL] [:= | DEFAULT expresie]
- constantele trebuie obligatoriu iniţializate, iar ulterior nu îşi vor putea schimba valoarea;
- variabilele NOT NULL trebuie obligatoriu iniţializate, iar ulterior nu vor putea primi valoarea NULL.

Exemple

```
v_functie VARCHAR2(9);
v_totalsal NUMBER(9,2):=0;
v_datainceput DATE:=sysdate+7;
c_taxa CONSTANT NUMBER(3,2):=8.25;
v_valid BOOLEAN NOT NULL:=TRUE;
v_pret produse.pret%TYPE:=100;
v_text CONSTANT VARCHAR2(30):='ABC';
v_stoc NUMBER DEFAULT NULL;
v_compusa produse%ROWTYPE;
```

5

Introducere în limbajul PL/SQL
Variabile non-PL/SQL

- **Variabile de mediu sau variabile de legătură ale aplicaţiilor gazdă (BIND VARIABLES)**
  - sunt variabile de legătură cu aplicaţia în care rulează motorul PL/SQL;
  - trebuie declarate în aplicaţie (în mediul gazdă) şi pot fi accesate şi modificate în cadrul blocurilor PL/SQL; după terminarea execuţiei blocului PL/SQL, variabila rămâne în mediul gazdă cu valoarea primită în urma rulării blocului (şi poate fi pasată altui bloc, realizând astfel transmiterea de valori între blocurile PL/SQL);
  - nu pot fi utilizate în cadrul procedurilor, funcţiilor sau pachetelor;
  - se declară în afara blocului PL/SQL prin cuvântului cheie VARIABLE (pentru declararea unei variabile numerice, nu se specifică precizia şi scala):

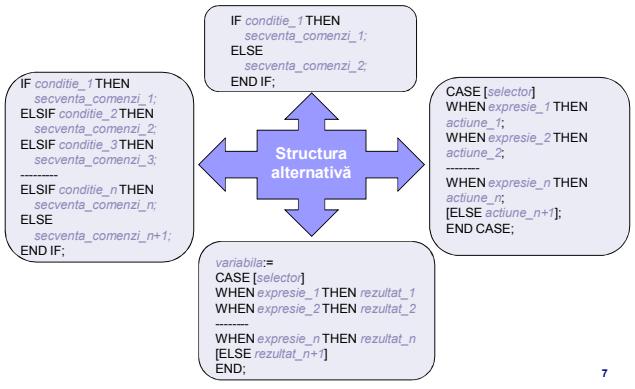
    VARIABLE g_numevariabilă TIP
  - pentru utilizarea lor în cadrul unui bloc PL/SQL sau într-o frază SQL din afara blocului se prefixează cu ":"

    :host_variabila:=v_variabila;
  - se afişează în afara blocului cu ajutorul comenzii PRINT (la afişare variabila nu se va prefixa cu ":")

    PRINT g_numevariabilă
- **Variabile de substituţie**
  - de regulă, variabilele de substituţie sunt folosite pentru a transmite valori spre comenzile SQL sau blocurile PL/SQL, în timp ce variabilele de legătură (*bind variables*) sunt folosite pentru a transmite valori în sens invers sau pentru a transfera valori între blocuri PL/SQL lansate succesiv (primul bloc setează variabila, următorul o consultă);
  - prin variabile de substituţie se pot transmite valori comenzilor SQL sau blocurilor PL/SQL lansate (folosind "&" sau "&&");
  - se pot invoca din comenzile SQL sau din blocurile PL/SQL prin **&nume_variabila** sau **&&nume_variabila**;
  - sunt locale sesiunii SQL în care au fost declarate;
  - variabilele de substituţie pot fi citite prin introducerea de valori de la tastatură (utilizând **ACCEPT nume_variabila**) sau se pot defini (prin **DEFINE nume_variabila = valoare**).

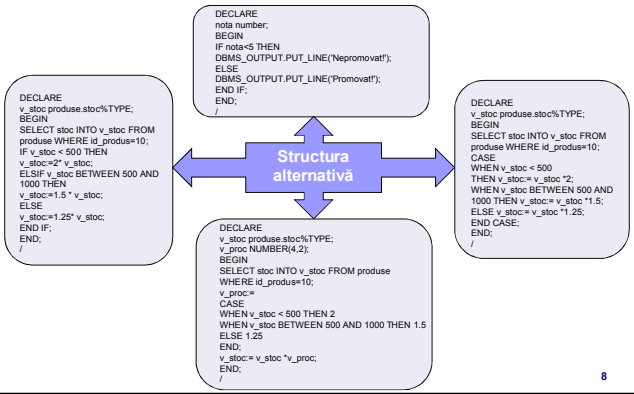
6

Elemente de programare procedurală
Structuri fundamentale de programare



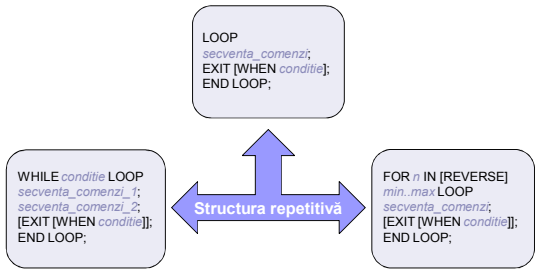
7

Elemente de programare procedurală
Structuri fundamentale de programare

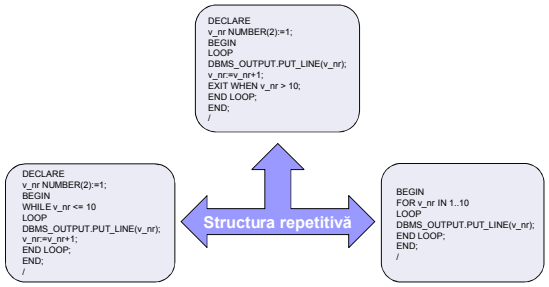


8

Elemente de programare procedurală
Structuri fundamentale de programare



Elemente de programare procedurală
Structuri fundamentale de programare



Mecanismul de cursor în SGBD ORACLE

- Atunci când se execută o comandă SQL, Oracle Server deschide o zonă de memorie (*context area*) în care comanda este executată. Cursorul este un pointer către această zonă
- În PL/SQL se utilizează două tipuri de cursoare:
  - implicit: declarat pentru toate instrucţiunile PL/SQL de tip LMD (INSERT/UPDATE/DELETE/SELECT care returnează o singură linie);
  - explicit: declarat şi gestionat de programator pentru a procesa individual fiecare linie returnată de o instrucţiune SELECT care returnează mai multe înregistrări.

Mecanismul de cursor în SGBD ORACLE
Cursorul implicit

- Atributele cursorului implicit, prin care se testează modul de execuţie a comenzilor LMD:
  - **SQL%ROWCOUNT**
  - **SQL%FOUND**
  - **SQL%NOTFOUND**

```
DECLARE
nu_exista_angajat EXCEPTION;
BEGIN
UPDATE angajati
SET comision=0.2*salariul
WHERE salariul < 2000;
IF SQL%NOTFOUND THEN
DBMS_OUTPUT.PUT_LINE('Nu exista angajati cu
salariul mai mic decat limita indicata');
ELSE
DBMS_OUTPUT.PUT_LINE('S-a modificat comisionul
pentru '|| SQL%ROWCOUNT || ' angajati');
END IF;
COMMIT;
END;
/
```

Mecanismul de cursor în SGBD ORACLE
Cursorul explicit

- Prelucrarea cursorului explicit presupune parcurgerea următoarelor etape:
  1. Declararea variabilelor
  2. Declararea cursorului, specificând fraza SELECT — `CURSOR nume_cursor IS SELECT .....;`
  3. Deschiderea cursorului — `OPEN nume_cursor;`
  4. Încărcarea liniilor din cursor în variabile — `FETCH nume_cursor INTO var1, var2, ...;`
  5. Închiderea cursorului — `CLOSE nume_cursor;`
- Atributele cursorului explicit:
  - *nume_cursor*%**ROWCOUNT**
  - *nume_cursor*%**FOUND**
  - *nume_cursor*%**NOTFOUND**
  - *nume_cursor*%**ISOPEN**

```
DECLARE
CURSOR c IS SELECT id_angajat, nume, salariul FROM angajati
WHERE id_departament=60;
v_id angajati.id_angajat%TYPE;
v_nume angajati.nume%TYPE;
v_sal angajati.salariul%TYPE;
BEGIN
OPEN c;
LOOP
FETCH c INTO v_id, v_nume, v_sal;
EXIT WHEN c%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('Salariatul '||v_nume||' are salariul: '||v_sal);
END LOOP;
CLOSE c;
END;
/
```

Mecanismul de cursor în SGBD ORACLE
Cursorul explicit

Gestiunea cursorului prin utilizarea unui ciclu FOR

FOR variabila_compusa IN nume_cursor LOOP

END LOOP;

- În acest caz, variabila compusă nu trebuie declarată;
- se realizează în mod implicit deschiderea, încărcarea şi închiderea cursorului.

```
DECLARE
CURSOR c IS SELECT id_angajat, nume, salariul
FROM angajati WHERE id_departament=60;
BEGIN
FOR rec_ang IN c LOOP
DBMS_OUTPUT.PUT_LINE('Salariatul '||
rec_ang.nume || ' are salariul: ' || rec_ang.salariul);
END LOOP;
END;
/
```

Utilizarea unui cursor direct în cadrul instrucţiunii FOR

FOR variabila_compusa IN (SELECT ...) LOOP

END LOOP;

- în acest caz cursorul nu este declarat, nu are nume, este reprezentat doar de interogarea SELECT din cadrul instrucţiunii FOR;
- dezavantajul este ca nu se pot utiliza atributele cursorului din cauza faptului că acesta nu are nume.

```
BEGIN
FOR rec_ang IN (SELECT id_angajat, nume,
salariul FROM angajati WHERE
id_departament=60)
LOOP
DBMS_OUTPUT.PUT_LINE('Salariatul '||
rec_ang.nume || ' are salariul: ' || rec_ang.salariul);
END LOOP;
END;
/
```


Gestiunea subprogramelor în SGBD ORACLE
Proceduri

- **Crearea procedurii:**

CREATE [OR REPLACE] PROCEDURE nume_proc
[(param1 [IN | OUT | IN OUT] TIP_DATA1, ...)]
-- tipul variabilelor este precizat fără dimensiune
(ex: NUMBER sau VARCHAR2)
IS | AS
-- zona de declarare a variabilelor utilizate
-- NU se utilizează DECLARE
BEGIN

[EXCEPTION]

END [nume_proc];

- **Apelul procedurii:**
  - prin nume dintr-un bloc PL/SQL anonim sau din alt subprogram;
  - prin apel direct cu **EXECUTE nume_proc** sau **CALL nume_proc**;
  - din alt mediu Oracle (ex. Oracle Developer Suite)

```
CREATE OR REPLACE PROCEDURE modifica_pret
( p_id IN produse.id_produs%TYPE, p_nr IN NUMBER )
AS
  v_nr_prod NUMBER;
BEGIN
  SELECT COUNT(*)
  INTO v_nr_prod
  FROM produse
  WHERE id_produs=p_id;

  IF v_nr_prod = 1 THEN
    UPDATE rand_comenzi
    SET pret = pret * p_nr
    WHERE id_produs=p_id;
    COMMIT;
  END IF;

EXCEPTION
  WHEN OTHERS THEN
    ROLLBACK;
END;
/

EXECUTE modifica_pret (100, 2);
```


Gestiunea subprogramelor în SGBD ORACLE
Funcţii

- **Crearea funcţiei:**

CREATE [OR REPLACE] FUNCTION nume_functie
[(param1 [IN] TIP_DATA1,...)]
RETURN TIP_DATA
-- tipul variabilelor este precizat fără dimensiune
(ex: NUMBER sau VARCHAR2)
IS | AS
-- zona de declarare a variabilelor utilizate în subprogram
-- nu se utilizează DECLARE
BEGIN

RETURN valoare;
[EXCEPTION]

END [nume_functie];

- **Apelul funcţiei:**
  - într-un bloc PL/SQL anonim sau în alt subprogram;
  - într-o comandă SQL

Atenţie! Funcţiile utilizate în expresii SQL nu trebuie să conţină comenzi LMD (update, delete, insert), comenzi LDD (create, alter, drop) şi nici comenzi pentru controlul tranzacţiilor (commit, rollback) şi nici nu trebuie să apeleze alte subprograme care să încalce aceste restricţii.

```
CREATE OR REPLACE FUNCTION vechime (p_id
angajati.id_angajat%TYPE) RETURN NUMBER
IS
v_vechime NUMBER;
BEGIN
SELECT ROUND((SYSDATE-data_angajare)/365) INTO
v_vechime
FROM angajati
WHERE id_angajat=p_id;
RETURN v_vechime;
EXCEPTION
WHEN NO_DATA_FOUND THEN RETURN 0;
WHEN OTHERS THEN RETURN -1;
END;
/

DECLARE
v_vechime NUMBER;
BEGIN
v_vechime:=vechime(100);
IF v_vechime>0 THEN
DBMS_OUTPUT.PUT_LINE(v_vechime);
ELSIF v_vechime=0 THEN DBMS_OUTPUT.PUT_LINE('Nu
exista angajatul!');
ELSE DBMS_OUTPUT.PUT_LINE('Eroare! - '||SQLERRM);
END IF;
END;
/
```

Exemple de teste grilă

Întrebarea 1

Se consideră tabela: *angajati (marca number(3), nume_angajat varchar2(35), data_angajarii date, functie varchar2(15), salariu number(5), id_departament number(3))*
Precizaţi care este efectul execuţiei următorului bloc PL/SQL:

```
DECLARE
CURSOR c_angajat IS SELECT salariu FROM angajati;
v_salariu angajati.salariu%TYPE;
BEGIN
OPEN c_angajat;
FETCH c_angajat INTO v_salariu;
CLOSE c_angajat;
FETCH c_angajat INTO v_salariu;
END;
/
```

a) secvenţa de program este eronată şi se va invoca excepţia INVALID_CURSOR
b) înregistrarea corespunzătoare primului angajat va fi parcursă de 2 ori
c) vor fi parcurse înregistrările corespunzătoare primilor 2 angajaţi
d) va fi invocată excepţia TOO_MANY_ROWS
e) blocul PL/SQL se va executa cu succes

Exemple de teste grilă

Întrebarea 2

Fie tabela: *produse (cod_produs number(3), denumire varchar2(20), pret number(5))*
Se dă funcţia PL/SQL:

```
CREATE OR REPLACE FUNCTION adauga_tva (p IN  number)
RETURN number IS
BEGIN
UPDATE produse SET pret=p*1.19;
RETURN (p);
END;
/
```

Considerând că tabela *produse* conţine minimum 100 de înregistrări, interogarea SQL-Oracle:

```
SELECT adauga_tva (pret) FROM produse;
```

a) afişează preţul produselor indexat cu 19%
b) afişează preţul iniţial al comenzilor
c) conţine o comandă de gestiune a tranzacţiilor
d) returnează o eroare deoarece funcţia creată conţine o comandă LMD incorect utilizată în cadrul cererii
e) afişează întotdeauna o singură valoare

Exemple de teste grilă

Întrebarea 3

Se consideră tabela: *produse (codp number(3), denp varchar2(20), um varchar2(3))*
Care dintre următoarele variante de specificare a parametrului unei proceduri PL/SQL nu este corectă:
a) (p_param IN VARCHAR2)
b) (p_param VARCHAR2)
c) (p_param VARCHAR2(50))
d) (p_param produse.denp%TYPE)
e) (p_param IN OUT VARCHAR2)

Întrebarea 4

In PL/SQL, care dintre urmatoarele atribute ale unui cursor explicit retine numarul total de inregistrari parcurse:
a) %ROWTYPE
b) %FOUND
c) %ROWCOUNT
d) %COUNT
e) %NOTFOUND

Exemple de teste grilă

Întrebarea 5

Se consideră tabela *Angajaţi* şi următoarea secvenţă de program PL/SQL:

```
DECLARE
CURSOR cursor1 IS SELECT id_angajat, nume FROM Angajati;
vid angajati.id_angajat%TYPE;
vnume CHAR (20);
BEGIN
OPEN cursor1;
LOOP
FETCH cursor1 INTO vid,vnume;
EXIT WHEN cursor1%NOTFOUND;
END LOOP;
END;
/
```

Care afirmaţie este corectă?
a)nu lucrează cu un cursor explicit
b)lucrează cu un cursor implicit
c)lucrează cu un cursor şi o tabelă virtuală
d)parcurge secvenţial o zonă de memorie internă
e)nu execută o structură repetitivă de program

Exemple de teste grilă

Întrebarea 6

În PL/SQL, dându-se tabelele PROD şi LUCRU, blocul:

```
BEGIN
UPDATE prod SET denp='mouse' WHERE codp=22;
IF SQL%NOTFOUND THEN
INSERT INTO lucru VALUES (0,'linie lipsa');
END IF;
END;
/
```

a) generează eroare
b) lucrează cu o viziune
c) conţine o structură repetitivă de program
d) lucrează cu un cursor explicit
e) conţine două operaţii de actualizare

Succes!