Baze de date-Anul 3 (semestrul 1)
Laborator 3 PL/SQL

Cursoare

Un cursor este o modalitate de a parcurge (linie cu linie) multimea de linii procesate returnate de o cerere 'multiple-row'. Această mulțime se numeşte *active set*.

Ø Cursoarele pot fi:
- implicite – care sunt declarate de PL/SQL in mod implicit pentru toate comenzile LMD si comanda SELECT, inclusiv comenzile care returneaza o singura linie.
- explicite – pentru cereri care returneaza mai mult de o linie, sunt definite cursoare explicite, denumite de programator si manipulate prin intermediul unor comenzi specifice.

Ø **Etapele** utilizarii unui cursor:
a) ***Declarare*** (în sectiunea declarativa a blocului PL/SQL):
CURSOR *c_nume_cursor* [*(parametru tip_de_Date, ..)*] *IS*
Comanda SELECT;

b) ***Deschidere*** (comanda OPEN), operatie ce identifică mulțimea de linii (*active set*):
OPEN *c_nume_cursor* [*(parametru, …)*]*;*

c) ***Incarcare*** (comanda FETCH). Numarul de variabile din clauza INTO trebuie sa se potriveasca cu lista SELECT returnata de cursor.
FETCH *c_nume_cursor* ***INTO*** *variabila, …;*

d) ***Verificare*** dacă nu am ajuns cumva la finalul mulțimii de linii folosind atributele:
C_nume_cursor%NOTFOUND – valoare booleana
C_nume_cursor%FOUND – valoare booleana

Daca nu s-a ajuns la final mergi la c).

e) ***Inchidere*** cursor (operatiune foarte importanta avand in vedere ca daca nu e inchis cursorul ramane deschis si consuma din resursele serverului, MAX_OPEN_CURSORS)
CLOSE *c_nume_cursor;*

Ø **Atributele** cursoarelor

Atribut	**Tip**	**Descriere**
%ISOPEN	Boolean	TRUE atunci când cursorul este deschis
%NOTFOUND	Boolean	TRUE dacă cea mai recentă operație FETCH nu a regăsit o linie
%FOUND	Boolean	TRUE dacă cea mai recentă operație FETCH a întors o linie
%ROWCOUNT	Number	Întoarce numărul de linii returnate până la momentul respectiv.

Ø **Clauza FOR UPDATE**
Comanda SELECT are urmatoarea extensie PL/SQL pentru blocarea explicita inregistrarilor ce urmeaza a fi prelucrate (modificate sau sterse):

SELECT …
FROM …
WHERE …
…
ORDER BY …
FOR UPDATE [OF lista_coloane] [NOWAIT | WAIT n];

- Daca liniile selectate de cerere nu pot fi blocate din cauza altor blocari atunci

- daca se foloseste NOWAIT este ridicata imediat eroarea ORA-00054
- daca nu se foloseste NOWAIT atunci se asteapta pana cand liniile sunt deblocate.
- daca se foloseste WAIT n atunci se asteapta un numar determinat de secunde inainte de a da eroare ca liniile ce trebuie selectate pentru modificare sunt blocate.

- Nu este recomandata anularea (ROLLBACK) sau permanentizarea schimbarilor inainte de a inchide cursorul ce foloseste FOR UPDATE pentru ca aceasta ar elibera blocarile realizate de acesta.
- Pentru a modifica o anumită linie returnata de un cursor se poate folosi clauza:

WHERE CURRENT OF nume_cursor

Aceasta clauza apare la finalul unei comenzi UPDATE si face referinta la un cursor care este deschis si s-a facut cel putin o incarcare din el (FETCH).

Exerciții: [Cursoare implicite]

1. Să se actualizeze liniile tabelului emp_pnu, mărind cu 10% valoarea comisionului pentru salariații având salariul mai mic decât o valoare introdusă de utilizator. Să se afişeze dacă au fost actualizate linii sau nu (SQL%FOUND), iar în caz afirmativ să se afişeze numărul de linii afectate (SQL%ROWCOUNT). Ce fel de cursor folosim?

2. Să se creeze un tabel DEP_EMP_PNU având câmpurile cod_dep şi cod_ang. Să se introducă într-o variabilă de tip tablou imbricat codurile departamentelor (în care există angajați), iar apoi, prin intermediul unei comenzi FORALL să se insereze aceste coduri şi codurile angajaților corespunzători în tabelul DEP_EMP_PNU. Pentru fiecare departamentm să se afişeze câți angajați au fost introduşi.

```
SET SERVEROUTPUT ON
DECLARE
  TYPE t_dep IS TABLE OF NUMBER;
  V_dep t_dep;
BEGIN
  SELECT department_id BULK COLLECT INTO v_dep FROM emp_pnu;
  FORALL j IN 1..v_dep.COUNT
    INSERT INTO dep_emp_pnu
      SELECT department_id,employee_id
      FROM   emp_pnu
      WHERE  department_id_id = v_dep(j);
  FOR j IN 1..v_dep.COUNT LOOP
    DBMS_OUTPUT.PUT_LINE ('Pentru departamentul avand codul ' ||
      V_dep(j) || ' au fost inserate ' ||
      SQL%BULK_ROWCOUNT(j)
      || inregistrari (angajati)');
  END LOOP;
  DBMS_OUTPUT.PUT_LINE ('Numarul total de inregistrari
     inserate este '||SQL%ROWCOUNT);
END;
/
SET SERVEROUTPUT OFF
```

Exerciții: [Introducere cursoare explicite]

3. Să se obțină câte o linie de forma ‘ <nume> are salariul anual <salariu annual> pentru fiecare angajat din departamentul 50. Se cer 4 soluții (WHILE, LOOP, FOR specific cursoarelor cu varianta de scriere a cursorului în interiorul său).

Soluția 1:

```
DECLARE
        CURSOR c_emp IS
                SELECT last_name, salary*12 sal_an
                FROM emp_pnu
```

```
                WHERE department_id = 50;
        V_emp c_emp%ROWTYPE;
BEGIN
        OPEN c_emp;
        FETCH c_emp INTO v_emp;
        WHILE (c_emp%FOUND) LOOP
                DBMS_OUTPUT.PUT_LINE (‘ Nume:’ || v_emp.last_name ||
                ‘ are salariul anual : ‘ || v_emp.sal_an);
                FETCH c_emp INTO v_emp;
        END LOOP;
        CLOSE c_emp;
END;
```

Soluția 2:

```
DECLARE
        CURSOR c_emp IS
                SELECT last_name, salary*12 sal_an
                FROM emp_pnu
                WHERE department_id = 50;
        V_emp c_emp%TYPE;
BEGIN
        OPEN c_emp;
        LOOP
                FETCH c_emp INTO v_emp;
                EXIT WHEN c_emp%NOTFOUND;
                DBMS_OUTPUT.PUT_LINE (‘ Nume:’ || v_emp.last_name ||
                  ‘ are salariul anual : ‘ || v_emp.sal_an);
        END LOOP;
        CLOSE c_emp;
END;
```

Soluția 3: // nu mai este nevoie explicit de OPEN, FETCH, CLOSE !!!

```
DECLARE
        CURSOR c_emp IS
                SELECT last_name, salary*12 sal_an
                FROM emp_pnu
                WHERE department_id = 50;
        BEGIN
                FOR v_emp IN c_emp LOOP
                        DBMS_OUTPUT.PUT_LINE (‘ Nume:’ || v_emp.last_name ||
                  ‘ are salariul anual : ‘ || v_emp.sal_an);
                END LOOP;
END;
```

Soluția 4:

```
BEGIN
  FOR v_rec IN  (SELECT last_name, salary*12 sal_an
              FROM   employees
              WHERE department_id = 50) LOOP
    DBMS_OUTPUT.PUT_LINE (‘ Nume:’ || v_rec.last_name ||
                ‘ are salariul anual : ‘ || v_rec.sal_an);
  END LOOP;
END;
/
```

4. Sa se afiseze salariatii care au salariul mai mic de 7000$, in urmatoarea forma:

   -Salariatul <nume> castiga <salariu>-.

5. Creați un bloc PL/SQL care determină cele mai mari n salarii, urmând paşii descrişi în continuare:
   a) creați un tabel top_salarii_pnu, având o coloană *salary*.
   b) Numărul n (al celor mai bine plătiți salariați) se va introduce de către utilizator (se va folosi comanda SQL*Plus ACCEPT şi o varianilă de substituție p_num).
   c) În secțiunea declarativă a blocului PL/SQL se vor declara 2 variabile: v_num de tip NUMBER (corespunzătoare lui p_num) şi v_sal de tipul coloanei salary. Se va declara un cursor emp_cursor pentru regăsirea salariilor în ordine descrescătoare (*se presupune că nu avem valori duplicate*).
   d) Se vor introduce cele mai mari mai bine plătiți n angajați în tabelul top_salarii_pnu;
   e) Afişați conținurtul tabelului top_salarii_pnu.
   f) Testați cazuri speciale, de genul n = 0 sau n mai mare decât numărul de angajați. Se vor elimina înregistrările din tabelul top_salarii_pnu după fiecare test.

Soluție:

```
ACCEPT p_num PROMPT ' … '
DECLARE
        V_num   NUMBER(3) := &p_num;
        V_sal     emp_pnu.salary%TYPE;
        CURSOR emp_cursor IS
                SELECT DISTINCT salary
                FROM    emp_pnu
                ORDER BY salary DESC;
BEGIN
        OPEN emp_cursor; -- folositi si alte variante de lucru cu cursorul !!!!
        FETCH emp_cursor INTO v_sal;
        WHILE emp_cursor%ROWCOUNT <= v_num AND emp_cursor%FOUND LOOP
                INSERT INTO top_salarii_pnu (salary)
                VALUES (v_sal);
                FETCH emp_cursor INTO v_sal;
        END LOOP;
        CLOSE emp_cursor;
END;
/
SELECT * FROM top_salarii_pnu;
```

Exerciții: [Cursoare cu parametru]

6. Să se declare un cursor cu un parametru de tipul codului angajatului, care regăseşte numele şi salariul angajaților având codul transmis ca parametru sau ale numele si salariile tuturor angajatilor daca valoarea parametrului este null. Să se declare o variabilă v_nume de tipul unei linii a cursorului. Să se declare două tablouri de nume (v_tab_nume), respectiv salarii (v_tab_sal). Să se parcurgă liniile cursorului în două moduri: regăsindu-le în v_nume sau în cele două variabile de tip tablou.

```
DECLARE
  CURSOR c_nume (p_id employees.employee_id%TYPE) IS
     SELECT last_name, salary
     FROM   employees
     WHERE p_id IS NULL  // conditia este adevarata pentru toate liniile tabelului atunci cand
                         // parametrul este null
    OR    employee_id = p_id;
  V_nume c_nume%ROWTYPE;
  -- sau
  /* TYPE t_nume IS RECORD
       (last_name employees.employee_id%TYPE,
        salary    employees.salary%TYPE
     v_nume t_nume;
  */
```

```
  TYPE t_tab_nume IS TABLE OF employees.last_name%TYPE;
  TYPE t_tab_sal  IS TABLE OF employees.salary%TYPE;
  V_tab_nume t_tab_nume;
  V_tab_Sal  t_tab_sal;
BEGIN
  IF c_nume%ISOPEN THEN
    CLOSE c_nume;
  END IF;
  OPEN c_nume (104);
  FETCH c_nume INTO v_nume;
  WHILE c_nume%FOUND LOOP
     DBMS_OUTPUT.PUT_LINE (‘ Nume:’ || v_nume.last_name ||
                    ‘ salariu : ‘ || v_nume.salary);
     FETCH c_nume INTO v_nume;
  END LOOP;
  CLOSE c_nume;
  -- eroare INVALID CURSOR
  --  FETCH c_nume INTO v_nume;
  DBMS_OUTPUT.PUT_LINE (‘ SI o varianta mai eficienta’);
  OPEN c_nume (null);
  FETCH c_nume BULK COLLECT INTO v_tab_nume, v_tab_sal;
  CLOSE c_nume;
  FOR i IN v_tab_nume.FIRST..v_tab_nume.LAST LOOP
     DBMS_OUTPUT.PUT_LINE (i || ‘ Nume:’ || v_tab_nume(i) ||
                  ‘ salariu : ‘ || v_tab_sal(i));
  END LOOP;
END;
/
```

7. Să se rezolve exercițiul 6 utilizând comanda LOOP.

```
DECLARE
  CURSOR c_nume (p_id employees.employee_id%TYPE) IS
     SELECT last_name, salary
     FROM   employees
     WHERE p_id IS NULL
    OR    employee_id = p_id;
  V_nume c_nume%ROWTYPE;
  -- sau
  /* TYPE t_nume IS RECORD
        (last_name employees.employee_id%TYPE,
         salary    employees.salary%TYPE
     v_nume t_nume;
  */
BEGIN
  OPEN c_nume (104);
  LOOP
    FETCH c_nume INTO v_nume;
    EXIT WHEN c_nume%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE (‘ Nume:’ || v_nume.last_name ||
                    ‘ salariu : ‘ || v_nume.salary);
  END LOOP;
  CLOSE c_nume;
END;
/
```

8. Să se rezolve exercițiul 6 folosind comanda FOR specifică lucrului cu cursoare.
Obs: La cursoare, comanda FOR realizează deschidere , incarcare si inchidere automata

```
DECLARE
  CURSOR c_nume (p_id employees.employee_id%TYPE) IS
     SELECT last_name, salary*12 sal_an
     FROM   employees
     WHERE p_id IS NULL
     OR    employee_id = p_id;
BEGIN
  FOR v_rec IN  c_nume (104) LOOP
    DBMS_OUTPUT.PUT_LINE (‘ Nume:’ || v_rec.last_name ||
                        ‘ salariu : ‘ || v_rec.sal_an);
  END LOOP;
END;
/
```

9. Utilizând un cursor parametrizat să se obțină codurile angajaților din fiecare departament şi pentru fiecare job. Rezultatele să fie inserate în tabelul *mesaje*, sub forma câte unui şir de caractere obținut prin concatenarea valorilor celor 3 coloane.

```
DECLARE
  v_cod_dep  departments.department_id%TYPE;
  v_cod_job  departments.job_id%TYPE;
  v_mesaj    VARCHAR2(75);
  CURSOR dep_job IS
    SELECT  department_id, job_id
    FROM    emp_pnu;
  CURSOR emp_cursor (v_id_dep NUMBER,v_id_job VARCHAR2) IS
    SELECT  employee_id || department_id || job_id
    FROM    emp_pnu
    WHERE   department_id = v_id_dep
    AND     job_id = v_id_job;
BEGIN
  OPEN dep_job;
  LOOP
    FETCH dep_job INTO v_cod_dep,v_cod_job;
    EXIT WHEN dep_job%NOTFOUND;
    IF emp_cursor%ISOPEN THEN
      CLOSE emp_cursor;
    END IF;
    OPEN emp_cursor (v_cod_dep, v_cod_job);
    LOOP
      FETCH emp_cursor INTO v_mesaj;
      EXIT WHEN emp_cursor%NOTFOUND;
      INSERT INTO mesaje (rezultat)
      VALUES (v_mesaj);
    END LOOP;
    CLOSE emp_cursor;
  END LOOP;
  CLOSE dep_job;
  COMMIT;
END;
/
```

Exerciții: [FOR UPDATE, WHERE CURRENT OF]

10. Să se dubleze valoarea salariilor angajaților înainte de 1 ianuarie 1995, care nu câştigă comision.

```
DECLARE
  CURSOR before95 IS
    SELECT *
    FROM   emp_pnu
```

```
   WHERE  commission_pct IS NULL
   AND    hire_date <= TO_DATE('01-JAN-1995','DD-MON-YYYY')
   FOR UPDATE OF salary NOWAIT;
BEGIN
  FOR x IN before95 LOOP
    UPDATE  emp_pnu
    SET     salary = salary*2
    WHERE CURRENT OF before95;
  END LOOP;
  COMMIT;  -- se permanentizeaza actiunea si se elibereaza blocarea
END;
/
```

11. Să se declare un cursor cu un parametru de tipul coloanei location_id, care determină departamentele din locația respectivă şi blochează liniile pe perioada prelucrării acestora. Să se deschidă cursorul folosind o variabilă de substituție pentru furnizarea parametrului. Să se actualizeze tabelul dep_pnu, dând valoarea 100 locației corespunzătoare liniei curente a cursorului.

```
DECLARE
 CURSOR c_test (p_loc departments.location_id%TYPE) IS
    SELECT 'x'
    FROM   dep_pnu
    WHERE location_id = &p_loc
     FOR UPDATE OF department_name NOWAIT;
  V_test c_test%ROWTYPE;
  -- sau
  -- v_test VARCHAR2(1);
BEGIN
  OPEN c_test (&loc);
  LOOP
    FETCH c_test INTO v_test;
    EXIT WHEN c_test%NOTFOUND;
    UPDATE dep_pnu
    SET  location_id = 100,
         -- aceasta coloana nu e blocata asa ca nu avem garantia
         -- daca va merge UPDATE-ul fara probleme sau se va bloca
         Department_name = 'x' -- acesta e selectat pentru update (blocare coloana)
    WHERE CURRENT OF c_test;
  END LOOP;
  CLOSE c_test;
   --ROLLBACK;
END;
/
```

Exerciții [Cursoare dinamice]

12. Să se declare un cursor dinamic care întoarce linii de tipul celor din tabelul emp_pnu. Să se citească o opțiune de la utilizator, care va putea lua valorile 1, 2 sau 3. Pentru opțiunea 1 deschideți cursorul astfel încât să regăsească toate informațiile din tabelul EMP_pnu, pentru opțiunea 2, cursorul va regăsi doar angajații având salariul cuprins între 10000 şi 20000, iar pentru opțiunea 3 se vor regăsi salariații angajați în anul 1990.

```
ACCEPT p_optiune PROMPT ‘Introduceti optiunea (1,2 sau 3) ‘
DECLARE
  TYPE  emp_tip IS REF CURSOR RETURN emp_pnu%ROWTYPE;
  V_emp emp_tip;
  V_optiune NUMBER := &p_optiune;
```

```
BEGIN
  IF v_optiune = 1 THEN
   OPEN v_emp FOR SELECT * FROM emp_pnu;
   --!!! Introduceți cod pentru afişare
  ELSIF v_optiune = 2 THEN
   OPEN v_emp FOR SELECT * FROM emp_pnu
               WHERE salary BETWEEN 10000 AND 20000;
   --!!! Introduceți cod pentru afişare
  ELSIF v_optiune = 3 THEN
   OPEN emp_pnu FOR SELECT * FROM emp_pnu
                WHERE TO_CHAR(hire_date, 'YYYY') = 1990;
   --!!! Introduceți cod pentru afişare
  ELSE
   DBMS_OUTPUT.PUT_LINE('Optiune incorecta');
  END IF;
END;
/
```

13. Să se citească o valoare n de la tastatura. Prin intermediul unui cursor deschis cu ajutorul unui şir dinamic să se regăsească angajații având salariul mai mare decât n. Pentru fiecare linie regăsită de cursor, dacă angajatul are comision, să se afişeze numele său şi salariul.

```
ACCEPT p_nr PROMPT 'Introduceți limita inferioara a salariului'

DECLARE
  TYPE  empref IS REF CURSOR;
  V_emp empref;
  v_nr  INTEGER := 100000;
BEGIN
  OPEN v_emp FOR
    'SELECT employee_id, salary FROM emp_pnu WHERE salary> :bind_var'
     USING v_nr;
 -- introduceti liniile corespunzatoare rezolvării problemei
END;
```

Exerciții: [Expresii cursor]

14. Să se listeze numele regiunilor şi pentru fiecare regiune să se afişeze numele țărilor. Se cer 2 metode de rezolvare (secvențial şi cu expresii cursor).

Varianta 1:

```
BEGIN
  FOR c_regiune IN (SELECT region_id, region_name
                FROM   regions)
  LOOP
    DBMS_OUTPUT.PUT_LINE (c_regiune.region_name);
    FOR c_tara IN (SELECT country_id, country_name
                FROM   countries
                WHERE  region_id =  c_regiune.region_id)
    LOOP
      DBMS_OUTPUT.PUT_LINE (c_tara.country_name);
    END LOOP;
  END LOOP;
END;
```

Varianta 2:

```
DECLARE
  CURSOR c_regiune IS
```

```
   SELECT region_name,
         CURSOR (SELECT country_name
              FROM   countries c
              WHERE  c.region_id = r.region_id)
   FROM   regions r;
  v_regiune   regions.region_name%TYPE;
  v_tara      SYS.REFCURSOR;
  TYPE tara_nume IS TABLE OF countries.country_name%TYPE
                INDEX BY BINARY_INTEGER;
  v_nume_tara  tara_nume;
BEGIN
  OPEN c_regiune;
  LOOP
   FETCH c_regiune INTO v_regiune, v_tara;
   EXIT WHEN c_regiune%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE (v_regiune);
   FETCH v_tara BULK COLLECT INTO v_nume_tara;
   FOR ind IN v_nume_tara.FIRST..v_nume_tara.LAST
    LOOP
     DBMS_OUTPUT.PUT_LINE (v_nume_tara (ind));
    END LOOP;
  END LOOP;
  CLOSE c_regiune;
END;
/
```