

### Cursoare

[Atributele cursoarelor](#)

[Clauza FOR UPDATE](#)

[Exerciții: \[Cursoare implicite\]](#)

[Exerciții: \[Introducere cursoare explicite\]](#)

[Exerciții: \[Cursoare cu parametru, ciclu cursor\]](#)

[Exerciții: \[FOR UPDATE, WHERE CURRENT OF\]](#)

[Exerciții: \[Cursoare dinamice\]](#)

[Exerciții: \[Expresii cursor\]](#)

### Cursoare

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

Cursoarele pot fi:

- implicite – care sunt declarate de PL/SQL în mod implicit pentru toate comenzile LMD și comanda SELECT, inclusiv comenzile care returnează o singură linie.
- explicite – pentru cereri care returnează mai mult de o linie, sunt definite cursoare explicite, denumite de programator și manipulate prin intermediul unor comenzi specifice.

**Etapele** utilizării unui cursor:

a) **Declarare** (în secțiunea declarativă a blocului PL/SQL):

```
CURSOR c_nume_cursor [ (parametru tip_de_Date, ..) ] IS  
Comanda SELECT;
```

b) **Deschidere** (comanda OPEN), operație ce identifică mulțimea de linii (active set):

```
OPEN c_nume_cursor [ (parametru, ...) ];
```

c) **Incarcare** (comanda FETCH ). Numărul de variabile din clauza INTO trebuie să se potrivească cu lista SELECT returnată 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 booleană

c\_nume\_cursor%FOUND – valoare booleană

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

e) **Inchidere** cursor (operațiune foarte importantă având în vedere că dacă nu e închis cursorul rămâne deschis și consumă din resursele serverului, MAX\_OPEN\_CURSORS)

```
CLOSE c_nume_cursor;
```

## Laborator 3 PL/SQL

---

### Atributele cursorurilor

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

### Clauza FOR UPDATE

Comanda SELECT are următoarea extensie PL/SQL pentru blocarea explicită a înregistrărilor ce urmează a fi prelucrate (modificate sau șterse):

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

- Dacă liniile selectate de cerere nu pot fi blocate din cauza altor blocari atunci
  - dacă se folosește NOWAIT este ridicată imediat eroarea ORA-00054,
  - dacă nu se folosește NOWAIT atunci se așteaptă până când liniile sunt deblocate,
  - dacă se folosește WAIT n atunci se așteaptă un număr determinat de secunde înainte de a da eroare ca liniile ce trebuie selectate pentru modificare sunt blocate.
- Nu este recomandată anularea (ROLLBACK ) sau permanentizarea schimbărilor înainte de a închide cursorul ce folosește FOR UPDATE pentru ca aceasta ar elibera blocările realizate de acesta.
- Pentru a modifica o anumită linie returnată de un cursor se poate folosi clauza:  
**WHERE CURRENT OF nume\_cursor**
- Aceasta clauza apare la finalul unei comenzi UPDATE și face referință la un cursor care este deschis și s-a făcut cel puțin o încărcare 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 afișeze dacă au fost actualizate linii sau nu (SQL%FOUND), iar în caz afirmativ să se afișeze numărul de linii afectate (SQL%ROWCOUNT). Ce fel de cursor folosim?

2. Să se creeze un tabel DEP\_EMP\_PNU având câmpurile cod\_dep și 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 și codurile angajaților

## Laborator 3 PL/SQL

---

corespunzători în tabelul DEP\_EMP\_PNU. Pentru fiecare departament să se afișeze câți angajați au fost introduși.

```
SET SERVEROUTPUT ON
CREATE TABLE dep_emp_pnu AS
SELECT department_id cod_dep, employee_id cod_ang
FROM employees
WHERE 1 = 0;

DECLARE
    TYPE t_dep IS TABLE OF NUMBER;
    v_dep t_dep;
BEGIN
    SELECT DISTINCT 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 = 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;
/

SELECT *
FROM dep_emp_pnu;

ROLLBACK;
SET SERVEROUTPUT OFF
```

### Exerciții: [Introducere cursoare explicite]

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

```
SET SERVEROUTPUT ON
--Soluția 1:
DECLARE
    CURSOR c_emp IS
    SELECT last_name, salary*12 sal_an
    FROM emp_pnu
    WHERE department_id = 50;
```

## Laborator 3 PL/SQL

---

```
        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%ROWTYPE;
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
```

## Laborator 3 PL/SQL

---

```
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\$, în următoarea formă:

'Salariatul <nume> castiga <salariu>'.

5. Creați un bloc PL/SQL care determină cele mai mari n salarii, urmând pașii descriși în continuare:

- creați un tabel top\_salarii\_pnu, având o coloană salary.
- Numărul n (al celor mai bine plătiți salariați) se va introduce de către utilizator (se va folosi comanda SQL\*Plus ACCEPT și o variabilă de substitutie p\_num).
- În secțiunea declarativă a blocului PL/SQL se vor declara 2 variabile: v\_num de tip NUMBER (corespunzătoare lui p\_num) și 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).
- Se vor introduce cei mai bine plătiți n angajați în tabelul top\_salarii\_pnu;
- Afișați conținutul tabelului top\_salarii\_pnu.
- 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.

```
CREATE TABLE top_salarii_pnu (salary NUMBER(8,2));  
SET VERIFY OFF  
ACCEPT p_num PROMPT 'Introduceti nr. n al celor mai bine platiti salariatii: '  
DECLARE  
    v_num NUMBER(8,2) := &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;  
ROLLBACK;
```

### Exerciții: [Cursoare cu parametru, ciclu cursor]

6. Să se declare un cursor cu un parametru de tipul codului angajatului, care regăsește numele și salariul angajaților având codul transmis ca parametru sau numele și salariile tuturor angajaților dacă 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;
```

## Laborator 3 PL/SQL

---

/

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 (**ciclu cursor**).

**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
        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.salary);
    END LOOP;
END;
/
```

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

## Laborator 3 PL/SQL

---

```
CREATE TABLE mesaje_pnu (rezultat VARCHAR2(100));

DECLARE
    v_cod_dep employees.department_id%TYPE;
    v_cod_job employees.job_id%TYPE;
    v_mesaj VARCHAR2(75);
    CURSOR dep_job IS
        SELECT DISTINCT department_id, job_id
        FROM employees;
    CURSOR emp_cursor (v_id_dep NUMBER, v_id_job VARCHAR2) IS
        SELECT employee_id || ', ' || department_id || ', ' || job_id
        FROM employees
        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_pnu (rezultat)
            VALUES (v_mesaj);
        END LOOP;
        CLOSE emp_cursor;
    END LOOP;
    CLOSE dep_job;
    COMMIT;
END;
/
SELECT *
FROM mesaje_pnu;
ROLLBACK;
```

### Exerciții: [FOR UPDATE, WHERE CURRENT OF]

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

```
DECLARE
    CURSOR before95 IS
        SELECT *
        FROM emp_pnu
        WHERE commission_pct IS NULL
        AND hire_date <= TO_DATE('01-IAN-1995', 'DD-MON-YYYY')
```



## Laborator 3 PL/SQL

---

```
        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;
/
SELECT *
FROM emp_pnu;
DELETE FROM emp_pnu;
INSERT INTO emp_pnu
SELECT *
FROM employees;
```

11. Să se declare un cursor cu un parametru de tipul coloanei `location_id`, care determină departamentele din locația respectivă și 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
    v_loc departments.location_id%TYPE:=&p_loc;
    CURSOR c_test (p_loc departments.location_id%TYPE) IS
        SELECT 'x'
        FROM dept_pnu
        WHERE location_id = v_loc
        FOR UPDATE OF department_name NOWAIT;
    v_test c_test%ROWTYPE;
    -- sau
    -- v_test VARCHAR2(1);
BEGIN
    OPEN c_test (v_loc);
    LOOP
        FETCH c_test INTO v_test;
        EXIT WHEN c_test%NOTFOUND;
        UPDATE dept_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;
/
```

## Laborator 3 PL/SQL

---

```
SELECT *  
FROM dept_pnu;
```

### 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 regsească toate informațiile din tabelul EMP\_pnu, pentru opțiunea 2, cursorul va regăsi doar angajații având salariul cuprins între 10000 și 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;  
    v_ang emp_pnu%ROWTYPE;  
BEGIN  
    IF v_optiune = 1 THEN  
        OPEN v_emp FOR SELECT * FROM emp_pnu;  
    ELSIF v_optiune = 2 THEN  
        OPEN v_emp FOR SELECT *  
                           FROM emp_pnu  
                           WHERE salary BETWEEN 10000 AND 20000;  
    ELSIF v_optiune = 3 THEN  
        OPEN v_emp FOR SELECT *  
                           FROM emp_pnu  
                           WHERE TO_CHAR(hire_date, 'YYYY') = 1990;  
    ELSE  
        DBMS_OUTPUT.PUT_LINE('Optiune incorecta');  
    END IF;  
    IF v_emp%ISOPEN THEN  
        LOOP  
            FETCH v_emp INTO v_ang;  
            EXIT WHEN v_emp%NOTFOUND;  
            DBMS_OUTPUT.PUT_LINE(v_ang.last_name);  
        END LOOP;  
        DBMS_OUTPUT.PUT_LINE('Au fost procesate ' || v_emp%ROWCOUNT  
                               || ' linii');  
        CLOSE v_emp;  
    END IF;  
END;  
/
```

13. Să se citească o valoare n de la tastatură. Prin intermediul unui cursor deschis cu ajutorul unui șir dinamic să se regsească angajații având salariul mai mare decât n. Pentru fiecare linie regăsită de cursor, dacă angajatul are comision, să se afișeze numele său și salariul.

## Laborator 3 PL/SQL

---

```
ACCEPT p_nr PROMPT 'Introduceți limita inferioara a salariului'
DECLARE
    TYPE empref IS REF CURSOR;
    v_emp empref;
    v_nr INTEGER := &p_nr;
    v_cod_ang emp_pnu.employee_id%TYPE;
    v_sal emp_pnu.salary%TYPE;
    v_com emp_pnu.commission_pct%TYPE;
BEGIN
    OPEN v_emp FOR
        'SELECT employee_id, salary, commission_pct
        FROM emp_pnu
        WHERE salary > :bind_var' USING v_nr;
    LOOP
        EXIT WHEN v_emp%NOTFOUND;
        FETCH v_emp INTO v_cod_ang, v_sal, v_com;
        IF v_com IS NOT NULL THEN
            DBMS_OUTPUT.PUT_LINE(v_cod_ang||' '||v_sal);
        ELSE
            DBMS_OUTPUT.PUT_LINE(v_cod_ang||' nu are comision.');
        END IF;
    END LOOP;
    CLOSE v_emp;
END;
/
```

### Exerciții: [Expresii cursor]

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

```
-- Varianta 1:
BEGIN
    FOR c_regiune IN (SELECT region_id, region_name
                      FROM regions) LOOP
        DBMS_OUTPUT.PUT_LINE (CHR(10)||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 (CHR(9)||c_tara.country_name);
        END LOOP;
    END LOOP;
END;
/
-- Varianta 2:
DECLARE
    CURSOR c_regiune IS
        SELECT region_name,
               CURSOR (SELECT country_name
```

## Laborator 3 PL/SQL

---

```

                                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_num IS TABLE OF countries.country_name%TYPE
        INDEX BY BINARY_INTEGER;
v_num_tara tara_num;
BEGIN
    OPEN c_regiune;
    LOOP
        FETCH c_regiune INTO v_regiune, v_tara;
        EXIT WHEN c_regiune%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE (CHR(10)||v_regiune);
        FETCH v_tara BULK COLLECT INTO v_num_tara;
        FOR ind IN v_num_tara.FIRST..v_num_tara.LAST LOOP
            DBMS_OUTPUT.PUT_LINE (CHR(9)||v_num_tara (ind));
        END LOOP;
    END LOOP;
    CLOSE c_regiune;
END;
/
```