**Cursoare -> vezi lab 3**

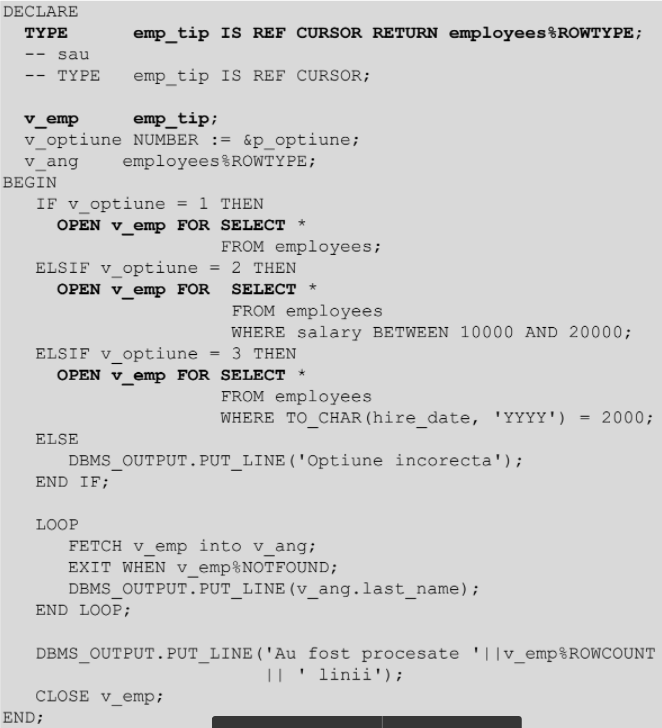
**= modalitate de a parcurge setul activ** (= o multime de linii (**=multiple-row**) intoarse de o cerere) **linie cu linie** => sunt necesare pt cererile care intorc mai mult de o linie

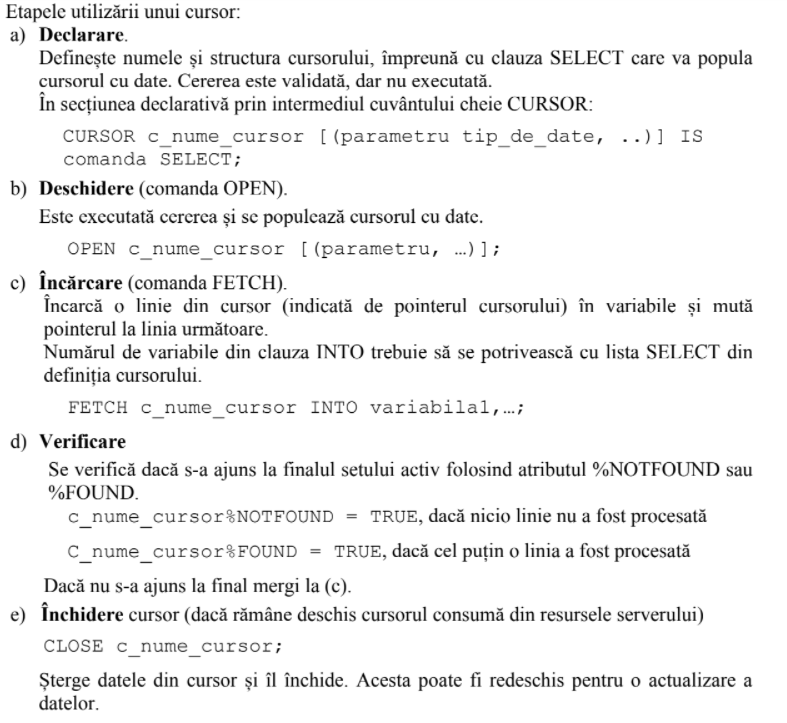
Definitia mai grea = pointer către o zonă de memorie (Private SQL Area) care stochează informații necesare pentru procesarea unei comenzi SELECT sau LMD

->la nivel de sesiune, ele există în memoria alocată sesiunii până la momentul încheierii acesteia

Exista:

* cursoare implicite / sql: inchis automat cand comanda se incheie
* cursoare explicite (def de utilizator): au specificat un nume asociat cu o comandă SELECT ce întoarce de obicei mai multe linii (si nu trb sa includa INTO).
* cursoare statice (= obiecte statice) = cursoare ale căror comenzi SQL sunt cunoscute la momentul compilării blocului.
* cursoare **dinamice** = variabile cursor = pointeri la un cursor
* li se pot asocia diferite cereri (coloanele obținute de fiecare cerere trebuie să corespundă declaraţiei variabilei cursor)
* trebuie declarate, deschise, încărcate şi închise în mod similar unui cursor static
* pot primi valori, o pot fi utilizate în expresii





Alte atribute utile:

- %ROWCOUNT reprezintă numărul de linii procesate;

- %ISOPEN este TRUE în cazul în care cursorul este deschis.

**Obs**:

Coloanele din cursor pot fi incarcate (la fetch) in cate o variabila de cate un tip diferit (corespunzator tipurilor returnate in select) SAU intr-o colectie (tablou, record) ce contine toate tipurile la un loc.

**Ex**:

DECLARE

--cu **cursor clasic**

cursor c1 is

select job\_title, job\_id

from jobs;

v\_titlu jobs.job\_title%type;

v\_id jobs.job\_id%type;

BEGIN

open c1;

loop

fetch c1 into v\_titlu, v\_id; -- daca fol tabel indexat trb bulk collect into

exit when c1%notfound; -- si asta arata mai bn asa

end loop;

close c1;

END;

**SAU:**

DECLARE

TYPE job IS RECORD

(titlu jobs.job\_title%TYPE,

id jobs.job\_id%TYPE,

nr\_ang number(2));

v\_job job;

CURSOR c1 IS

SELECT job\_title, job\_id

FROM jobs;

BEGIN

OPEN c1;

LOOP

FETCH c1 INTO v\_job;

IF c1%NOTFOUND THEN EXIT; END IF;

END LOOP;

CLOSE c1;

END;

**Tipuri de cursoare**

* **ciclu cursor** -> trebuie doar declarat (operaţiile de deschidere, încărcare şi închidere fiind implicite). ex:

DECLARE

CURSOR c IS

SELECT department\_name nume, COUNT(employee\_id) nr FROM departments d, employees e

WHERE d.department\_id=e.department\_id(+) GROUP BY department\_name;

BEGIN

FOR i in c LOOP

IF i.nr=0 THEN DBMS\_OUTPUT.PUT\_LINE('In departamentul '|| i.nume|| ' nu lucreaza angajati');

ELSIF i.nr=1 THEN DBMS\_OUTPUT.PUT\_LINE('In departamentul '|| i.nume || ' lucreaza un angajat');

ELSE DBMS\_OUTPUT.PUT\_LINE('In departamentul '|| i.nume|| ' lucreaza '|| i.nr||' angajati');

END IF;

END LOOP;

END;

/

* **ciclu cursor cu subcereri** -> nu e necesara nici macar declararea. ex: direct

BEGIN

FOR i in (SELECT department\_name nume, COUNT(employee\_id) nr

FROM departments d, employees e

WHERE d.department\_id=e.department\_id(+)

GROUP BY department\_name) LOOP

IF i.nr=0 THEN DBMS\_OUTPUT.PUT\_LINE('In departamentul '|| i.nume|| ' nu lucreaza angajati');

ELSIF i.nr=1 THEN DBMS\_OUTPUT.PUT\_LINE('In departamentul '|| i.nume || ' lucreaza un angajat');

ELSE DBMS\_OUTPUT.PUT\_LINE('In departamentul '|| i.nume|| ' lucreaza '|| i.nr||' angajati');

END IF;

END LOOP;

END;

/

* **cursor parametrizat** -> comenda SELECT depinde de parametri ce **pot** fi modificaţi la momentul execuţiei. ex:

--cursor simplu

DECLARE

v\_x number(4) := &p\_x; ...

CURSOR c (paramentru NUMBER) IS

SELECT department\_name nume, COUNT(employee\_id) nr

FROM departments d, employees e WHERE d.department\_id=e.department\_id GROUP BY department\_name

HAVING COUNT(employee\_id)> paramentru;

BEGIN

OPEN c(v\_x);

LOOP FETCH c INTO ...; EXIT WHEN c%NOTFOUND; ... END LOOP;

CLOSE c;

END;

/

--ciclu cursor

DECLARE

v\_x number(4) := &p\_x;

CURSOR c (paramentru NUMBER) IS

SELECT department\_name nume, COUNT(employee\_id) nr

FROM departments d, employees e WHERE d.department\_id=e.department\_id GROUP BY department\_name

HAVING COUNT(employee\_id)> paramentru;

BEGIN

FOR i in c(v\_x) LOOP ... END LOOP;

END;

/

--ciclu cursor cu subcerere

DECLARE

v\_x number(4) := &p\_x;

BEGIN

FOR i in (SELECT department\_name nume, COUNT(employee\_id) nr

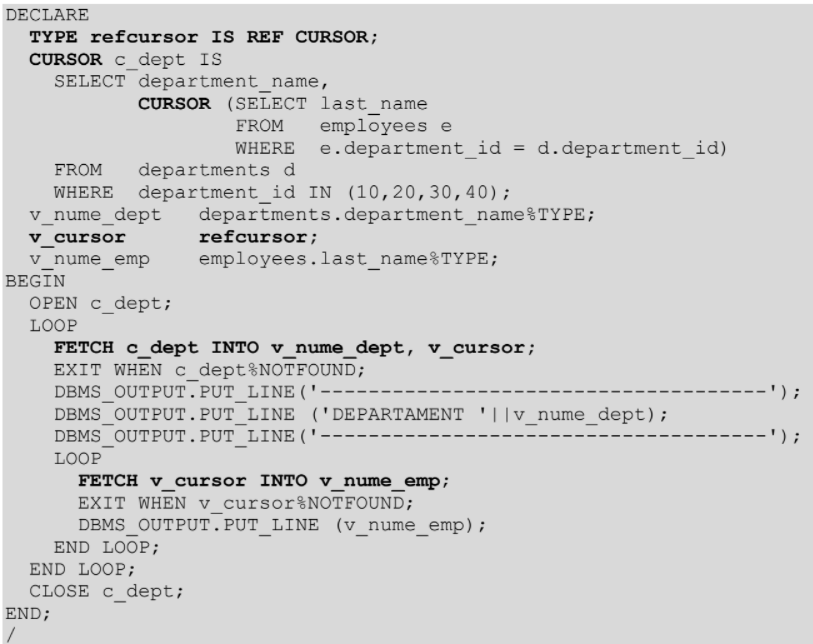
FROM departments d, employees e WHERE d.department\_id=e.department\_id GROUP BY department\_name

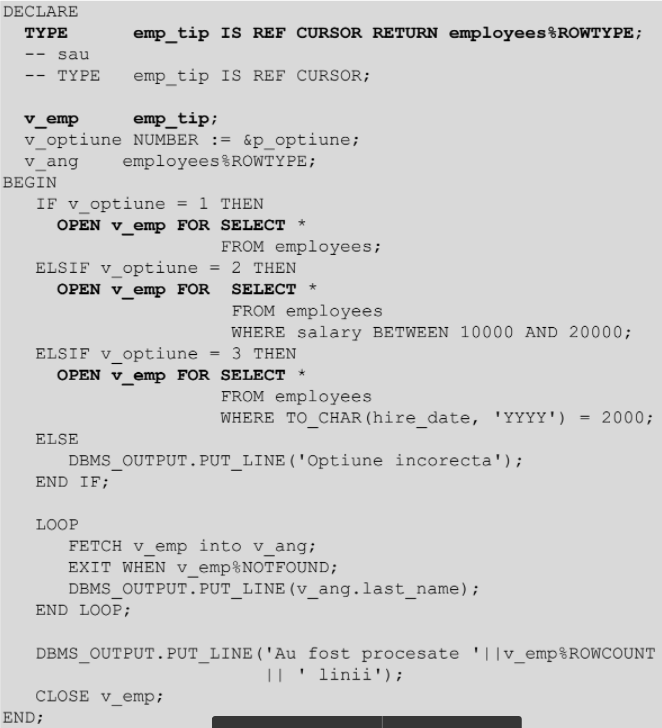
HAVING COUNT(employee\_id)> v\_x)

LOOP ... END LOOP;

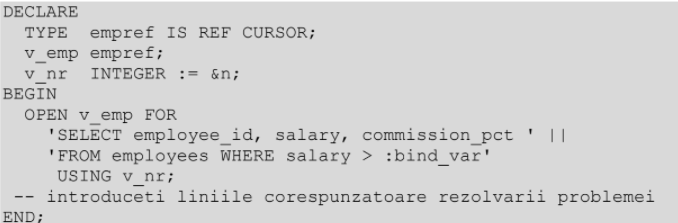
END;

* **expresii cursor = nested cursor** ->





* **cursor dinamic** ->
* **cursor** **deschis cu ajutorul unui șir dinamic** ->



**Stergerea / Actualizare liniilor**

* necesita blocarea lor inainte
* cu FOR UPDATE in comanda select din deschiderea cursorului + NOWAIT / WAIT N pt astepta n secunde sau a nu astepta deblocarea lor atunci cand liniile sunt deja blocate din alta parte

- dacă se folosește NOWAIT apare 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 pentru ca liniile ce trebuie selectate pentru modificare să fie deblocate.

DECLARE

CURSOR c IS SELECT \* FROM emp\_\*\*\* WHERE TO\_CHAR(hire\_date, 'YYYY') = 2000

**FOR UPDATE OF salary NOWAIT;**

BEGIN

FOR i IN c LOOP

UPDATE emp\_\*\*\* SET salary= salary+1000 **WHERE CURRENT OF c**; // pt a modif liniile intoarse de cursor

END LOOP;

END;

/