**CURSORUL ÎN PL/SQL**

* 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
    - **explicit**: declarat şi gestionat de programator.

**CURSORUL IMPLICIT**

* Este declarat de PL/SQL implicit pentru toate comenzile de manipulare a datelor (INSERT, UPDATE, DELETE, SELECT);
* Dacă o instrucțiune LMD nu afectează nici o linie a tabelei, nu se generează automat o excepție tip NO\_DATA\_FOUND;
* Atributele cursorului implicit:
  + **SQL%ROWCOUNT**
  + **SQL%FOUND**
  + **SQL%NOTFOUND**
  + **SQL%ISOPEN este mereu FALSE**
* Atributele se referă la cea mai recentă instrucțiune LMD. Înaintea primei instrucțiuni LMD din bloc, toate atributele au valoarea NULL;
* După o instrucțiune COMMIT sau ROLLBACK, SQL%ROWCOUNT are valoarea 0.

**Atenție** - Cursorul implicit NU este util pentru instrucțiunea SELECT folosită cu INTO. Aceasta va rula cu succes doar dacă interogarea returnează un singur rând și atunci SQL%ROWCOUNT va fi 1, SQL%FOUND va fi TRUE iar SQL%NOTFOUND va fi FALSE. Dacă interogarea nu returnează nici un rând sau returnează două sau mai multe, se va ridica o excepție (NO\_DATA\_FOUND sau TOO\_MANY\_ROWS).

**Exemplu**

Se șterg produsele din categoria *hardware3* care nu au fost comandate. Se afișează numărul de rânduri şterse.

SET SERVEROUTPUT ON

BEGIN

DELETE FROM produse p

WHERE categorie='hardware3' and not exists (select 1 from rand\_comenzi r where p.id\_produs=r.id\_produs);

DBMS\_OUTPUT.PUT\_LINE (SQL%ROWCOUNT || ' randuri sterse');

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE (SQL%ROWCOUNT || ' randuri afectate');

END;

/

După ROLLBACK, atributul SQL%ROWCOUNT devine 0. Rezultatul rulării este:

*10 randuri sterse*

*0 randuri afectate*

Se încearcă adăugarea unei regiuni și apoi modificarea denumirii produsului având codul 3. În cazul în care acest produs nu există (comanda update nu realizează nici o modificare) va fi afişat un mesaj corespunzător.

BEGIN

INSERT INTO regiuni VALUES(5,'Oceania');

UPDATE produse

SET denumire\_produs='cafea'

WHERE id\_produs=3;

IF SQL%NOTFOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista produsul cu acest cod');

END IF;

ROLLBACK;

END;

/

Observație: dacă regiunea cu id-ul 5 există deja, se va declanșa o excepție, dacă nu, se va rula comanda UPDATE. Atributul SQL%NOTFOUND se referă la ultima comandă, în acest caz, la UPDATE.

Se șterge din tabela REGIUNI, regiunea a cărei ID este introdus de utilizator prin intermediul variabilei de substituție *g\_rid*. Mesajul este afișat folosind variabila de mediu *nr\_sters*.

ACCEPT g\_rid PROMPT 'Introduceti id-ul regiunii:'

VARIABLE nr\_sters varchar2(100)

DECLARE

BEGIN

DELETE FROM regiuni WHERE id\_regiune=&g\_rid;

:nr\_sters:=TO\_CHAR(SQL%ROWCOUNT)||' INREGISTRARI STERSE';

END;

/

PRINT nr\_sters

ROLLBACK;

**CURSORUL ÎN PL/SQL**

**CURSORUL EXPLICIT**

* se foloseşte pentru a procesa individual fiecare linie (înregistrare) returnată de o instrucţiune SELECT ce returnează mai multe înregistrări.
* mulţimea înregistrărilor returnate de o instructiune SELECT este numită *mulţime rezultat*.
* cursorul păstrează un pointer către linia curentă în cadrul unei mulţimi rezultat.

Verificarea stării unui cursor explicit se realizează prin intermediul următoarelor atribute:

* + **NUME\_CURSOR%ISOPEN -** evaluat la TRUE în cazul în care cursorul este deschis;
  + **NUME\_CURSOR%NOTFOUND** - evaluat la TRUE în cazul în care cel mai recent FETCH nu a returnat nici o linie;
  + **NUME\_CURSOR%FOUND** - complementul lui %NOTFOUND;
  + **NUME\_CURSOR%ROWCOUNT** - are ca valoare numărul liniilor returnate până în momentul curent.

**PARCURGEREA FOLOSIND OPEN-LOOP-FECTH-CLOSE**

In această abordare, prelucrarea cursorului explicit presupune parcurgerea următoarelor etape:

* + - 1. se declară variabilele în care vor fi încărcate valorile corespunzătoare unei linii din cursor;
      2. se declară cursorul explicit, specificându-se un nume pentru acesta şi definindu-se interogarea de procesat în cadrul lui:

***DECLARE nume\_cursor IS SELECT........................;***

* + - 1. se deschide cursorul prin intermediul instrucţiunii *OPEN*, care execută interogarea şi legarea tuturor variabilelor referite. Înregistrările returnate de interogare sunt desemnate drept *set activ de date*, care pot fi de acum încărcate.

***OPEN nume\_cursor;***

* + - 1. utilizându-se instrucţiunea FETCH, se încarcă linia curentă din cursor în variabile. Fiecare încărcare determină mutarea pointerului cursorului la linia următoare din setul activ de date.

***FETCH nume\_cursor INTO var1, var2,..............;***

* + - 1. este închis cursorul prin instructiunea *CLOSE*, care dezafectează setul activ de linii. Cursorul poate fi din nou deschis pentru a stabili un nou set activ de linii.

***CLOSE nume\_cursor;***

Pentru a procesa liniile unui cursor explicit se defineşte de obicei o buclă pentru executarea unui FETCH în fiecare iteraţie. În final, toate liniile din setul activ sunt procesate şi un FETCH executat fără succes poziţionează atributul %NOTFOUND pe TRUE.

Înaintea primului FETCH, %NOTFOUND se evaluează la NULL, ca şi în cazul în care FETCH nu se execută niciodată cu succes.

**Exemple**:

Să se afişeze lista cu numele şi salariul angajaţilor din departamentul 60 folosind un cursor explicit și trei variabile scalare:

set serveroutput on

DECLARE

cursor ang\_cursor is select id\_angajat, nume, salariul from angajati where id\_departament=60;

ang\_id angajati.id\_angajat%type;

ang\_nume angajati.nume%type;

ang\_sal angajati.salariul%type;

BEGIN

dbms\_output.put\_line('Lista cu salariariile angajatilor din departamentul 60');

open ang\_cursor;

loop

fetch ang\_cursor into ang\_id, ang\_nume, ang\_sal;

exit when ang\_cursor%notfound;

dbms\_output.put\_line('Salariatul '||ang\_nume||' are salariul: '||ang\_sal);

end loop;

close ang\_cursor;

end;

/

Pentru o flexibilitate mai mare se poate utiliza o variabilă de tip înregistrare pentru încărcarea valorilor din cursor. Această variabilă de tip record poate avea aceleaşi atribute ca şi cursorul prin specificarea proprietăţii %ROWTYPE. În acest caz încărcarea din cursor se va face direct prin instrucţiunea *fech var\_cursor into var\_record*.

Exemplul de mai sus poate fi rescris astfel:

set serveroutput on

declare

cursor ang\_cursor is select id\_angajat, nume, salariul from angajati where id\_departament=60;

--tipul record definit cu %ROWTYPE pt incarcarea valorilor cursorului

ang\_rec ang\_cursor%rowtype;

begin

dbms\_output.put\_line('Lista cu salariariile angajatilor din departamentul 60');

open ang\_cursor;

loop

fetch ang\_cursor into ang\_rec;

exit when ang\_cursor%notfound;

dbms\_output.put\_line('Salariatul '||ang\_rec.nume||' are salariul: '||ang\_rec.salariul);

end loop;

close ang\_cursor;

end;

/

Să se încarce în tabela MESAJE primii 5 angajaţi (id şi nume)

CREATE TABLE mesaje

(cod varchar2(7),

nume varchar2(20)

);

DECLARE

v\_id angajati.id\_angajat%type;

v\_nume angajati.nume%type;

**CURSOR c1 IS SELECT id\_angajat, nume FROM angajati;**

BEGIN

**OPEN c1;**

FOR i IN 1..5 LOOP

**FETCH c1 INTO v\_id, v\_nume;**

INSERT INTO mesaje VALUES(v\_id, v\_nume);

END LOOP;

**CLOSE c1;**

END;

/

SELECT \* FROM mesaje;

Testul de ieşire din buclă în acest caz se poate face şi cu ajutorul atributului %ROWCOUNT. Tabela mesaje nu are cheie primară deci pot fi adăugate aceleași rânduri de mai multe ori.

delete from mesaje;

DECLARE

v\_id angajati.id\_angajat%type;

v\_nume angajati.nume%type;

**CURSOR c1 IS SELECT id\_angajat, nume FROM angajati;**

BEGIN

**OPEN c1;**

LOOP

**FETCH c1 INTO v\_id, v\_nume;**

EXIT WHEN **c1%ROWCOUNT**>5 OR **c1%NOTFOUND**;

INSERT INTO mesaje VALUES (v\_id, v\_nume);

END LOOP;

**CLOSE c1;**

END;

/

SELECT \* FROM mesaje;

**PARCURGEREA FOLOSIND FOR-LOOP**

**FOR nume\_record IN nume\_cursor LOOP**

**--------------------------------------------------------**

**END LOOP;**

În acest caz, tipul RECORD nu trebuie declarat. Se realizează în mod implicit deschiderea, încărcarea și închiderea cursorului. **Atenție**, nu deschideți explicit cursorul înainte de FOR sau închideți cursorul după END LOOP, se va declanșa o excepție.

**Exemplu**:

Se afişează printr-un ciclu FOR numele şi salariile angajaţilor din departamentul 60:

set serveroutput on

declare

cursor ang\_cursor is select id\_angajat, nume, salariul from angajati where id\_departament=60;

begin

dbms\_output.put\_line('Lista cu salariariile angajatilor din departamentul 60');

for ang\_rec in ang\_cursor loop

dbms\_output.put\_line('Salariatul '||ang\_rec.nume||' are salariul: '||ang\_rec.salariul);

end loop;

end;

/

**PARCURGEREA FOLOSIND FOR-LOOP A UNUI CURSOR INLINE (ANONIM)**

În acest caz cursorul nu este declarat, nu are nume, este reprezentat doar de interogarea SELECT din cadrul instrucţiunii FOR, astfel:

**FOR NUME\_RECORD IN (SELECT......) LOOP**

**.....................................................................................**

**END LOOP;**

Dezavantajul în acest caz este ca nu se pot utiliza atributele cursorului din cauza faptului că acesta nu are nume.

**Exemplu:**

Să se afişeze suma aferentă salariilor din fiecare departament:

set serveroutput on

declare

begin

dbms\_output.put\_line('Total salarii pe fiecare departament:');

for dep\_rec in

(select d. id\_departament dep, sum(a.salariul) sal

from angajati a, departamente d

where a.id\_departament=d.id\_departament

group by d.id\_departament)

loop

dbms\_output.put\_line('Departamentul '||dep\_rec.dep||' are de platit salarii in valoare de: '||dep\_rec.sal||' RON');

end loop;

end;

/

**UTILIZAREA CURSORULUI CU PARAMETRU**

Pentru o flexibilitate mai mare se pot declara şi utiliza cursori cu parametru care transmit valorile parametrilor actuali în cererile SQL. Declararea cursorului cu parametru se face astfel:

**CURSOR NUME\_CURSOR (PARAMETRU1 TIP\_DATA,.....)**

**IS SELECT .................;**

*Deschidere:*

**OPEN NUME\_CURSOR(VALOARE\_PARAMETRU1,......);**

Cursoarele parametrizate nu oferă o funcţionalitate suplimentară ci doar o modalitate simplă şi clară de a specifica valori de intrare.

Tipurile parametrilor sunt scalare, dar nu li se precizează dimensiunea; ele fiind referite în interogare.

**Exemple**:

Să se afişeze produsele al căror cantitate totală comandată este mai mare decât o valoare primită drept parametru.

SET SERVEROUTPUT ON

DECLARE

**CURSOR** **c\_prod (p\_val NUMBER)** IS

SELECT p.id\_produs, p.denumire\_produs, sum(r.cantitate) total

FROM produse p, rand\_comenzi r

WHERE p.id\_produs =r.id\_produs

GROUP BY p.id\_produs, p.denumire\_produs

HAVING sum(r.cantitate)>p\_val

ORDER BY total desc;

v\_val NUMBER(5);

rec\_prod c\_prod%rowtype;

BEGIN

v\_val:=500;

DBMS\_OUTPUT.PUT\_LINE('Produsele al caror cantitate vândută este mai mare decat '|| v\_val);

IF NOT c\_prod%ISOPEN THEN

OPEN **c\_prod (v\_val)**;

END IF;

LOOP

FETCH c\_prod into rec\_prod;

EXIT WHEN c\_prod%notfound;

DBMS\_OUTPUT.PUT\_LINE('Din produsul '||rec\_prod.id\_produs||', '||rec\_prod.denumire\_produs||', s-au vandut ' ||rec\_prod.total||' unitati');

END LOOP;

CLOSE c\_prod;

END;

/

Să se afişeze pentru fiecare comanda produsele comandate. În acest caz se utilizează două variabile de tip cursor. Vom folosi parcurgerea cursorului cu FOR:

SET SERVEROUTPUT ON

DECLARE

--cursorul care va prelua comenzile incheiate

CURSOR c\_com IS

SELECT nr\_comanda, data

FROM comenzi

Where modalitate= 'online'

ORDER BY nr\_comanda;

--cursorul care, pentru fiecare comanda, va afisa produsele din cadrul acesteia, ordonate descrescator

CURSOR c\_prod (p\_nr\_comanda NUMBER) IS

SELECT r.id\_produs, p.denumire\_produs, r.cantitate

FROM produse p, rand\_comenzi r

WHERE p.id\_produs=r.id\_produs

AND r.nr\_comanda=p\_nr\_comanda

ORDER BY r.id\_produs desc;

BEGIN

FOR rec\_com in c\_com LOOP

DBMS\_OUTPUT.PUT\_LINE('Comanda '|| rec\_com.nr\_comanda ||' incheiata la data de '||rec\_com.data);

FOR rec\_prod in c\_prod(rec\_com.nr\_comanda) LOOP --cursorul primeste drept parametru numarul comenzii care a fost afisata

DBMS\_OUTPUT.PUT\_LINE('Din produsul '||rec\_prod.id\_produs||', '||rec\_prod.denumire\_produs||', s-au comandat ' ||rec\_prod.cantitate||' bucati');

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('============');

END LOOP;

END;

/

**CLAUZA FOR UPDATE**

* Se adaugă clauza FOR UPDATE în interogarea asociată cursorului pentru a bloca liniile afectate atunci când cursorul este deschis;
* Când mai multe tabele sunt implicate în interogare, se poate folosi FOR UPDATE pentru a impune blocarea liniilor unei tabele anume. Liniile unei tabele sunt blocate numai în cazul în care clauza FOR UPDATE face o referire la o coloană din acea tabelă;
* Se blochează setul de înregistrări ale cursorului în 2 variante: **NOWAIT** şi **WAIT n**:

**CURSOR C IS SELECT .... FROM....**

**FOR UPDATE [OF COLUMN\_NAME] [NOWAIT|WAIT n];**

* Clauza NOWAIT - determină apariția unei excepții dacă rândurile sunt blocate de o altă sesiune;
* Clauza WAIT n – așteaptă, dacă este cazul, n secunde deblocarea rândurile. Dacă după expirarea secundelor rândurile sunt tot blocate va declanșa o excepție.

**Exemplu:**

Se creează tabela Situatie care pastreaza informatii despre comenzi: codul, valoarea comenzii. Se adaugă în aceasta coloana TVA, care va păstra valoarea TVA pentru fiecare comandă. Se creează un cursor căruia i se adaugă clauza FOR UPDATE pentru a bloca liniile afectate din tabelă, atunci când cursorul este deschis, iar pentru fiecare comandă din cursor se va calcula valoarea TVA.

DROP TABLE situatie;

CREATE TABLE situatie AS

SELECT c.nr\_comanda cod, SUM(r.cantitate\*r.pret) as valoare

FROM comenzi c, rand\_comenzi r

WHERE c. nr\_comanda =r. nr\_comanda

AND c.modalitate= 'online'

GROUP BY c. nr\_comanda;

ALTER TABLE situatie

ADD(tva NUMBER(10));

DECLARE

CURSOR c\_situatie IS

SELECT cod, valoare, tva

FROM situatie

FOR UPDATE OF tva NOWAIT;

BEGIN

FOR rec\_situatie IN c\_situatie LOOP

UPDATE situatie

SET tva=valoare\*0.19

WHERE cod=rec\_situatie.cod;

DBMS\_OUTPUT.PUT\_LINE('Comanda '||rec\_situatie.cod||' are valoarea totala de '||rec\_situatie.valoare||' RON si TVA de: '||rec\_situatie.tva );

END LOOP;

END;

/

SELECT \* FROM situatie;

**Atenţie**: În cazul de mai sus se poate observa faptul că atributul *REC\_SITUATIE.TVA* este NULL după execuţia comenzii *UPDATE* deoarece nu se actualizează automat şi cursorul odată cu tabela. Acesta trebuie închis şi redeschis pentru a fi vizibile actualizările din tabelă.

Pentru manipularea cât mai uşoară a comenzilor LMD UPDATE şi DELETE se poate utiliza clauza **WHERE CURRENT OF** care permite actualizarea înregistrărilor pe baza liniei curente din cursor.

**UPDATE tabela**

**SET camp=....**

**WHERE CURRENT OF nume\_cursor;**

* Se poate referi linia din tabela originară, pentru actualizare sau ştergere, prin intermediul liniei curente a cursorului (cea procesată de ultima instrucţiune FETCH);
* Clauza FOR UPDATE trebuie inclusă în definiţia cursorului pentru a bloca liniile în momentul execuţiei instrucţiunii OPEN.

**Exemplu:**

Exemplu de mai sus poate fi rescris, actualizarea înregistrărilor din tabela SITUATIE realizându-se cu clauza **WHERE CURRENT OF:**

DROP TABLE situatie;

CREATE TABLE situatie AS

SELECT c.nr\_comanda cod, SUM(r.cantitate\*r.pret) as valoare

FROM comenzi c, rand\_comenzi r

WHERE c. nr\_comanda =r. nr\_comanda

AND c.modalitate= 'online'

GROUP BY c. nr\_comanda;

ALTER TABLE situatie

ADD(tva NUMBER(10));

DECLARE

CURSOR c\_situatie IS

SELECT cod, valoare, tva

FROM situatie

FOR UPDATE OF tva NOWAIT;

BEGIN

FOR rec\_situatie IN c\_situatie LOOP

UPDATE situatie

SET tva=valoare\*0.19

WHERE CURRENT OF c\_situatie;

DBMS\_OUTPUT.PUT\_LINE('Comanda '||rec\_situatie.cod||' are valoarea totala de '||rec\_situatie.valoare||' RON si tva de: '||rec\_situatie.tva );

END LOOP;

END;

/

SELECT \* FROM situatie;

**Exerciţii:**

1. Afișați informații despre primele 3 comenzi care au cea mai mare valoare.
2. Afişaţi informaţii despre primii 5 salariaţi angajaţi (se va realiza filtrarea în funcție de câmpul Data\_Angajare).