9. CURSOARE

Obiective.

În acest Capitol, vom învăţa despre:

Manipularea cursoarelor.

Folosirea Cursor FOR Loops şi Nesting Cursors.

Cursoare sunt zone de memorie care ne permit să alocam o zonă de memorie şi să accesam informaţia provenită dintr-o instrucţiune SQL. De exemplu, se foloseste un cursor pentru a opera cu toate liniile tabelului STUDENT pentru acei studenţi care iau parte la un anumit curs (au asociate intrări în tabelul inscrisi) (ENROLLMENT). În acest capitol, vom învăţa să declaram un cursor explicit care activează un utilizator pentru a prelucra mai multe linii returnate de un query şi care permit utilizatorului să scrie codul care va prelucra fiecare linie linie unul câte unul.

9.1 Manipularea cursoarelor

Obiective.

Folosirea Record Types.
Prelucrarea unui Cursor Explicit.
Folosirea Cursor Attributes.
Toate la un loc.

Pentru ca Oracle să proceseze o instrucţiune SQL, este nevoie ca el să creeze o zonă de memorie cunoscută ca buffer de context; Această va conţine informaţia necesară prelucrării instrucţiunilor.

Această informaţie include numărul de linii care au fost prelucrate de instrucţiune, un pointer la reprezentarea instrucţiunii analizate(analizarea unei instrucţiuni SQL este procesul prin care informaţia este transferată la server, timp în care instrucţiunea SQL este evaluată ca validă). Într-un query, setul activ se referă la liniile care vor fi returnate.

Un cursor este un handler, sau pointer, la suprafaţa de context. Prin cursor, un program PL SQL poată să controleze suprafaţa de context şi ce se întâmpla cu aceasta atunci când instrucţiunea este prelucrată.

Două caracteristici importante ale cursorului sunt după cum urmează:

1. Cursoare nepermit să aducem şi să prelucram liniile care sunt returnate de o instrucţiune SELECT, una câte una.
2. Un cursor se numeşte aşa pentru ca el să poată să fie referit.

Tipuri de Cursore.

Există două tipuri de cursoare:

1.Un cursor implicit este în mod automat declarat de Oracle de fiecare dată când o instrucţiune SQL este executată. Utilizatorul nu va fi conştient că acest lucru se întâmpla şi nu va fi în stare să controleze sau să prelucreze informaţia dintr-un cursor implicit.

2. Un cursor explicit este definit de program pentru oricare query care returnează mai mult de un linie de informaţii. Aceasta înseamnă că mediul de programare a declarat cursorul într-un bloc scris în PL/SQL. Această declaraţie permite ca aplicaţia să proceseze secvenţial fiecare linie de informaţii aşa cum este acum returnată de cursor.

Cursor Implicit.

Pentru o mai bună înţelegere a posibilităţilor unui cursor explicit, trebuie mai întâi să trecem în revistă procesul unui cursor implicit.

Orice bloc de PL SQL este un cursor implicit oricând se execută o instrucţiune SQL, atât timp cât un cursor explicit nu este declarat pentru acea instrucţiune SQL.

Un cursor este în mod automat asociat cu o instrucţiune DML(Data Manipulation) (UPDATE, DELETE, INSERT).

Toate instrucţiunile de ACTUALIZARE şi ŞTERGERE au cursoare care identifică setul de linii care vor fi afectate de operaţia respectivă.

O instrucţiune INSERT are nevoie de un loc de pentru a primi informaţiile care urmează să să fie inserate în baza de date; Cursorul implicit realizează această nevoie.

Cursor implicit este intitulat cursorul 'SQL %'.

Procesarea Cursorului Impliucit

Cursorul implicit este folosit pentru a procesa instrucţiuni INSERT, UPDATE, DELETE, şi SELECT INTO. În timpul procesării unui cursor implicit, Oracle execută în mod automat operaţiile OPEN, FETCH, şi CLOSE.

Un cursor implicit nu poată să vă spună câte linieurile au fost afectate de un update. SQL%ROWCOUNT returnează numărul de linieuri care au fost actualizate. El poate să se întrebuinţeze după cum urmează:

```
SET SERVEROUTPUT ON
BEGIN
   UPDATE studenţi
    SET nume = 'Banu'
   WHERE nume LIKE 'B%';
   DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT);
END;
```

Se Consideră următorul exemplu al unui cursor implicit.

```
SET SERVEROUTPUT ON;
DECLARE
  v_nume      VARCHAR2(35);
  v_prenume VARCHAR2(35);
BEGIN
  SELECT nume, prenume
    INTO v_nume, v_prenume
    FROM studenţi
    WHERE student_id = 123;
  DBMS_OUTPUT.PUT_LINE (' nume Student: '||
     v_nume ||' '|| v_prenume);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
     DBMS_OUTPUT.PUT_LINE
     (nu exista student cu ID 123');
END;
```

Este important de notat că Oracle asociază în mod automat un cursor implicit cu o instrucţiune SELECT INTO şi aduce valori pentru variabilele

v_nume şi v_prenume. Odată ce instrucţiunea SELECT INTO este terminată, Oracle închide cursorul implicit.

Spre deosebire de cursorul implicit, cursor explicit este definit de un program pentru oricare query care returnează mai mult decât un linie de informaţii.

Astfel un cursor explicit se defineste după cum urmează. Mai întâi se declară cursorul, apoi se deschide cursorul declarat, Apoi aduceţi cursorul ?? declarat şi deschis de mai devreme. În cele din urmă se închide cursorul.

Cursor Explicit.

Singura modalitate de generare ale unui cursor explicit este ca acesta să fie declarat în secţiunea DECLARE a unui blocul PL SQL.

Avantajele de a declara un cursor explicit în locul unui cursor indirect implicit sunt acelea că, cursorul explicit dă mai mult control programarii din mediul de programare. Cursoare implicite sunt mai puţin eficiente decât cursoarele explicite, şi astfel este mai greu de "a prinde în cursă" erorile informaţiilor.

Procesul de lucru cu un cursor explicit constă să următorii paşii:

1. Declararea cursorului. Acesta iniţializează cursorul în memorie.
2. Deschiderea cursorului. Cursorul declarat anterior poate acum să fie deschis; Memoria este alocată.
3. Aducerea cursorului. Cursorul anterior declarat şi deschis poate acum să recupereze informaţii; Acesta este procesul de aducere a cursorului.
4. Închiderea cursorului. Cursorul anterior declarat, deschis, şi adus trebuie acum închis pentru eliberarea memoriei alocate.

Declararea unui Cursor.

Declararea unui cursor defineşte numele cursorului şi îl asociează cu o instrucţiune SELECT. Primul pas este de a declara cursorul folosind următoarea sintaxă:

CURSOR c-nume-cursor
IS instrucţiune SELECT

Convenţiile de nume care se întrebuinţează în Oracle recomandă să numim întotdeauna un cursor c_numecursor. Folosind litera c_la începutul numelui ne va fi întotdeauna clar că numele se referă la un cursor.
Nu este posibilă folosirea unui cursor fără execuţia completă a secvenţei:

(1) declare,
(2) deschidere,
(3) aducere, şi în cele din urmă
(4) închidere.

Pentru a explica aceşti patru paşi, următorul exemplu va avea fragmentele de cod pentru fiecare pas şi în cele din urmă ne va arăta procesul complet.

Acesta este un fragment PL SQL care demonstrează pasul întâi de declarare al unui cursor. Un cursor numit C_mycursor este declarat ca o instrucţiune SELECT a tuturor liniilor din tabelul zipcode care are articolul state egal cu 'NY'.

De exemplu.

```
DECLARE
  CURSOR C_MyCursor IS
    SELECT *
     FROM zipcode
     WHERE stat = 'RO';
...
<aici va continua codul cu deschiderea, aducerea şi închiderea cursorului>
```

Numele cursoarelor satisfac aceleaşi reguli ca cele ale numelor simbolice din PL SQL. Pentru că numele cursorului este un nume simbolic din PL SQL, el trebuie să fie declarat înainte ca el să fie referit. Orice instrucţiune SELECT validă poate să fie întrebuinţată pentru a defini un cursor, care includ comenzile de reuniune UNION sau diferenta MINUS.

Tipuri de Înregistrări.

O înregistrare este o structură de informaţie, care înseamnă că este formată din mai mult decât un element. Înregistrările sunt foarte mult similare cu o linie a unui table dintro baza de date, dar fiecare element al înregistrării nu depinde de el însuşi. PL SQL are definite trei tipuri de înregistrări:

(1) bazată pe tabel (table-based),
(2) bazată pe cursor (cursor-based),
(3) definite de programator (programmer-defined).

O înregistrare bazată pe tabel(table-based) este o înregistrare a cărei structură este dată de lista de coloane a tabelului.

O înregistrare bazată pe cursor(cursor-based) este o înregistrare a cărei structură se atribuie elementele unui cursor definit anticipat. Pentu a crea o înregistrare bazată pe tabel sau bazată pe cursor, se foloseste atributul %ROWTYPE.

< nume-inregistrare> < nume-tabel or nume-cursor >%ROWTYPE

De exemplu:

```
SET SERVEROUTPUT ON
DECLARE
  vr_student studenti%ROWTYPE;
BEGIN
  SELECT *
    INTO vr_student
    FROM studenti
   WHERE student_id = 156;
  DBMS_OUTPUT.PUT_LINE (vr_student. nume||' '
    ||vr_student.prenume||' are  ID 156');
EXCEPTION
  WHEN no_data_found
    THEN
```

```
            RAISE_APPLICATION_ERROR(-2001,' Studenti '||
             ' nu este  in    basa de date');
    END;
```

Variabila vr_student este de tip înregistrare din tabelul studenti al bazei de date care există deja. Aceasta înseamnă că ea are aceleaşi componente cao linie din tabelul studenti.

O înregistrare bazată pe cursor este cam acelaşi lucru, cu excepţia faptului că este definită de lista select a cursorului declart.

Când facem referire la elementele înregistrării, folosiţi aceeaşi sintaxă pe care o folosim la tabele.

Nume-inregistrare. Nume-element

Pentru a defini o variabilă care se bazează pe o înregistrare a unui cursor, cursorul trebuie mai întâi să fie declarat. În următorul laborator, veţi porni prin a declara un cursor şi după aceea veţi continua procesul de a deschide cursorul, de a-l aduce, şi în cele din urmă închiderea cursoruli.

O înregistrare bazată pe tabel este definită dintr-o structură particulară a unui tabel. Consideram următorul fragment de cod:

```
DECLARE
  vr_zip ZIPCODE%ROWTYPE;
  vr_instructor Instructor%ROWTYPE;
```

Înregistrarea vr_zip are structura asemănătoare cu o linie a tabelului ZIPCODE. Elementele lui sunt ORAS, STAT, şi ZIP.

Este important de notat faptul că coloana ORAS a tabelului ZIPCODE a fost definită ca VARCHAR2(15), atributul ORAS al înregistrării vr_zip va avea aceeaşi structură de tip de date (datatype).

Similar, înregistrarea vr_instructor se bazează pe o linie a tabelului instructor.

9.2 Folosirea Cursoarelor FOR Loops şi Nesting

Obiectivul acestui paragraf este sa invatam cum sa folosim un Cursor FOR Loop si Să procesam Cursoare Nested

Există o metodă alternativă de a manipula cursoare aceasta se numeşte cursor FOR loop din cauza sintaxei care se întrebuinţează. Când folosim cursorul FOR loop, procesul de deschidere, aducere, şi închidere este manipulat în mod implicit. Acesta face blocul mult mai simplu de a fi codificat şi mai uşor de întreţinut.

Cursorul FOR Loop specifică o succesiune de comenzi ce se repetată odată pentru fiecare linie întoarsă de cursor. Folosim cursorul FOR loop dacă avem nevoie de să ADUCEM şi SĂ PROCESAM fiecare înregistrare a cursorului.

De exemplu.
Presupunem că există un tabel intitulat log cu o singură coloană.

```
create table table_log
      (description VARCHAR2(250));

DECLARE
  CURSOR c_student IS
    SELECT id_student, name, prenume
     FROM studenti
    WHERE id_student < 110;
BEGIN
  FOR r_student IN c_student
  LOOP
    INSERT INTO table_log
      VALUES(r_student.nume);
  END LOOP;
END;
```

Următorul exemplu este al unui cursor nested. Trecem în revistă codul.
De exemplu

```
SET SERVEROUTPUT ON
DECLARE
  v_taxa              course.taxa%TYPE;
  v_id_instructor  instructor.id_instructor%TYPE;
  CURSOR c_inst IS
    SELECT nume, prenume, id_instructor
     FROM instructor;
  CURSOR c_taxa IS
    SELECT c.taxa
     FROM curs c, sectie s, inscrisi e
     WHERE s.id_instructor = v_id_instructor
      AND c.curs_no = s.curs_no
      AND s.id_sectie = e.id_sectie;
BEGIN
  FOR r_inst IN c_inst
  LOOP
     v_id_instructor := r_id_inst.instructor;
     v_ taxa := 0;
     DBMS_OUTPUT.PUT_LINE(
       ' taxa generata de instructor '||
       r_inst.nume||' '||r_inst.prenume
       ||' is');
     FOR r_cost IN c_ taxa
     LOOP
       v_taxa := v_taxa + NVL(r_cost.cost, 0);
     END LOOP;
     DBMS_OUTPUT.PUT_LINE
     ('     '||TO_CHAR(v_ taxa,'$999,999'));
  END LOOP;
END;
```