

Baze de date

- Lucrare de laborator -

CURSOARE

Obiective

1. Manipularea cursoroanelor
2. Declararea unui cursor
3. Folosirea Record Types
4. Folosirea Cursor Attributes
5. Prelucrarea unui Cursor Explicit, Implicit, Cursor FOR Loops și Nesting Cursors

1. Introducere

Cursori sunt zone de memorie care ne permit să alocăm o zonă de memorie și să accesăm informația provenită dintr-o instrucțiune SQL. De exemplu, se folosește un cursor pentru a opera cu toate liniile tabelului STUDENT pentru acei studenți care iau parte la un anumit curs (au asociate intrări în tabelul înscriși) (ENROLLMENT). În acest capitol, vom învăța să declarăm un cursor explicit care activează un utilizator pentru a prelucra mai multe linii returnate de un query și care permit utilizatorului să scrie codul care va prelucra fiecare linie dintr-unul.

2. Manipularea cursoroanelor

Un cursor este un handler, sau pointer, la suprafața de context. Prin cursor, un program PL SQL poate să controleze suprafața de context și ce se întâmplă cu aceasta atunci când instrucțiunea este prelucrată. Această informație include numărul de linii care au fost prelucrate de instrucțiune, un pointer la reprezentarea instrucțiunii analizate (analizarea unei instrucțiuni SQL este procesul prin care informația este transferată la server, timp în care instrucțiunea SQL este evaluată ca validă). Într-un query, setul activ se referă la liniile care vor fi returnate. Pentru ca Oracle să proceseze o instrucțiune SQL, este nevoie ca el să creeze o zonă de memorie cunoscută ca buffer de context; Această va conține informația necesară prelucrării instrucțiunilor.

O caracteristică importantă a cursoroanelor este că ne permit să aducem și să prelucrăm liniile care sunt returnate de o instrucțiune SELECT, una câte una.

3. Declararea unui Cursor

Declararea unui cursor definește numele cursorului și îl asociază cu o instrucțiune SELECT. Primul pas este de a declara cursorul folosind următoarea sintaxă:

CURSOR c-nume-cursor
IS instructiune SELECT

Convențiile de nume care se întrebuintează 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 execuția completă a secvenței:

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

Pentru a explica acești patru pași, următorul exemplu va avea fragmentele de cod pentru fiecare pas și 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 instrucțiune 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 și închiderea cursorului>

Numele cursoroarelor satisfac aceleași 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 instrucțiune SELECT validă poate să fie întrebuintată pentru a defini un cursor, care includ comenzile de reuniune UNION sau diferența MINUS.

4. Tipuri de Cursore

Există două tipuri de cursoare:

1. Un *cursor implicit* este în mod automat declarat de Oracle de fiecare dată când o instrucțiune SQL este executată. Utilizatorul nu va fi conștient că acest lucru se întâmplă și nu va fi în stare să controleze sau să prelucreze informația dintr-un cursor implicit.

2. Un *cursor explicit* este definit de program pentru oricare query care returnează mai mult de un linie de informații. Aceasta înseamnă că mediul de programare a declarat cursorul într-un bloc scris în PL/SQL. Această declarație permite ca aplicația să proceseze secvențial fiecare linie de informații așa cum este acum returnată de cursor.

4.1. Cursor Implicit

Pentru o mai bună înțelegere a posibilităților 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 instrucțiune SQL, atât timp cât un cursor explicit nu este declarat pentru acea instrucțiune SQL.

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

Toate instrucțiunile de ACTUALIZARE și ȘTERGERE au cursoare care identifică setul de linii care vor fi afectate de operația respectivă.

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

Cursor implicit este intitulat cursorul 'SQL %'.

Procesarea Cursorului Implicit

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

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

```
SET SERVEROUTPUT ON
BEGIN
    UPDATE studenți
    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 studenți
```

```

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 instrucțiune SELECT INTO și aduce valori pentru variabilele `v_nume` și `v_prenume`. Odată ce instrucțiunea 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 informații.

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

4.2. Cursor Explicit

Singura modalitate de generare ale unui cursor explicit este ca acesta să fie declarat în secțiunea 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 puțin eficiente decât cursoarele explicite, și astfel este mai greu de “a prinde în cursă” erorile informațiilor.

Procesul de lucru cu un cursor explicit constă în următorii pași:

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

5. Tipuri de Înregistrări

O înregistrare este o structură de informație, care înseamnă că este formată din mai mult decât un element. Înregistrările sunt foarte mult similare cu o linie a unui table

dintr-o bază de date, dar fiecare element al înregistrării nu depinde de el însuși. 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. Pentru a crea o înregistrare bazată pe tabel sau bazată pe cursor, se folosește 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 baza 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 aceleași componente cao linie din tabelul studenti.

O înregistrare bazată pe cursor este cam același lucru, cu excepția faptului că este definită de lista select a cursorului declarat.

Când facem referire la elementele înregistrării, folosiți aceeași 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, veți porni prin a declara un cursor

și după aceea veți continua procesul de a deschide cursorul, de a-l aduce, și în cele din urmă închiderea cursorului.

O înregistrare bazată pe tabel este definită dintr-o structură particulară a unui tabel. Considerăm 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`, și `ZIP`. Este important de notat faptul că în coloana `ORAS` a tabelului `ZIPCODE` a fost definită ca `VARCHAR2(15)`, atributul `ORAS` al înregistrării `vr_zip` va avea aceeași structură de tip de date (datatype). Similar, înregistrarea `vr_instructor` se bazează pe o linie a tabelului `instructor`.

6. Folosirea cursoroanelor FOR Loops și Nesting

Obiectivul acestui paragraf este să învățăm cum să folosim un Cursor FOR Loop și să procesăm Curseuri Nested

Există o metodă alternativă de a manipula curseuri aceasta se numește cursor FOR loop din cauza sintaxei care se întrebuițează. Când folosim cursorul FOR loop, procesul de deschidere, aducere, și închidere este manipulat în mod implicit. Acesta face blocul mult mai simplu de a fi codificat și mai ușor de întreținut.

Cursorul FOR Loop specifică o succesiune de comenzi ce se repetă odată pentru fiecare linie întoarsă de cursor. Folosim cursorul FOR loop dacă avem nevoie de să ADUCEM și 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:

```

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, inscriși 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_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;

```