CUPRINS

5. *PL/SQL* – Gestiunea cursoarelor

- Un cursor este un pointer către o zonă de memorie (*Private SQL Area*) care stochează informații necesare pentru procesarea unei comenzi *SELECT* sau *LMD*.

În acest capitol se discută cursoarele la nivel sesiune.

- Cursoarele la nivel de sesiune:
  - există în memoria alocată sesiunii până la momentul încheierii acesteia.
- Vizualizarea *V$OPEN_CURSOR* oferă informații despre cursoarele deschise la nivel de sesiune ale fiecărei sesiuni utilizator.

În continuare, din motive de simplificare a exprimării, pentru un cursor la nivel de sesiune se va utiliza termenul de cursor.

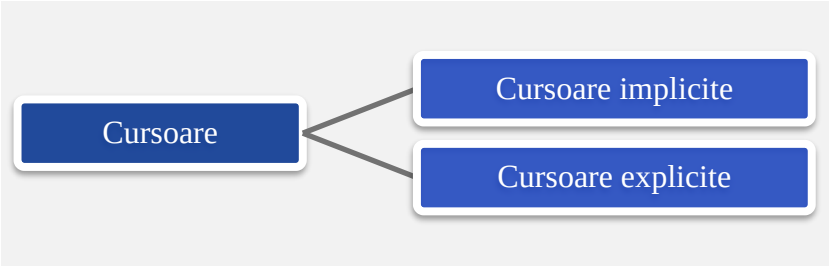

Fig. 5.1. Tipuri de cursoare

- Categorii de cursoare:
  - cursoare implicite
    - construite și gestionate automat de *PL/SQL*
  - cursoare explicite
    - construite și gestionate de către utilizatori.
- Atributele care furnizează informații despre cursoare:
  - pot fi referite doar de comenzi procedurale
  - pot fi referite utilizând sintaxa
    - pentru cursoarele implicite

      `SQL%nume_atribut`
    - pentru cursoarele explicite

      `nume_cursor%nume_atribut`

- lista atributelor:
  - *%ROWCOUNT*
    - este de tip întreg (*PLS_INTEGER*);
    - are valoarea *NULL* dacă nu a fost rulată nicio comandă *SELECT* sau *LMD*;
    - reprezintă numărul liniilor întoarse de ultima comandă *SELECT* sau numărul de linii afectate de ultima comandă *LMD*;
    - dacă numărul de linii este mai mare decât valoarea maximă permisă de tipul *PLS_INTEGER* (2.147.483.647), atunci întoarce o valoare negativă.
  - *%FOUND*
    - este de tip boolean;
    - are valoarea *NULL* dacă nu a fost rulată nicio comandă *SELECT* sau *LMD*;
    - în cazul cursoarelor implicite are valoarea *TRUE* dacă ultima comandă *SELECT* a întors cel puţin o linie sau ultima comandă *LMD* a afectat cel puțin o linie;
    - în cazul cursoarelor explicite are valoarea *TRUE* dacă ultima operaţie de încărcare (*FETCH*) dintr-un cursor a avut succes.
  - *%NOTFOUND*
    - este de tip boolean;
    - are semnificaţie opusă faţă de cea a atributului *%FOUND*.
  - *%ISOPEN*
    - este de tip boolean;
    - indică dacă un cursor este deschis;
    - în cazul cursoarelor implicite, acest atribut are întotdeauna valoarea *FALSE*, deoarece un cursor implicit este închis de sistem imediat după execuția instrucţiunii *SQL* asociate.
  - *%BULK_ROWCOUNT*
    - vezi în Capitolul 4 comanda *FORALL*
  - *%BULK_EXCEPTIONS*
    - vezi în Capitolul 4 comanda *FORALL*

5.1. Cursoare implicite

- *PL/SQL* deschide automat un cursor implicit la nivel de sesiune de fiecare dată când este rulată o comandă *SELECT* sau *LMD*.
- Mai sunt denumite și cursoare *SQL*.
- Cursorul implicit este închis automat, atunci când comanda se încheie.
- Valorile atributelor asociate cursorului rămân disponibile până când este rulată o altă comandă *SELECT* sau *LMD*.

`Exemplul 5.1` **`- vezi curs`**

- Atributul *SQL%NOTFOUND* nu este util în cazul comenzii *SELECT INTO*.
- Dacă această comandă nu întoarce linii, atunci apare imediat excepția *NO_DATA_ FOUND* (înainte să se poată verifica valoarea atributului *SQL%NOTFOUND*).
- Dacă în lista *SELECT* a comenzii se utilizează funcții agregat, atunci este întoarsă întotdeauna o valoare. În acest caz, valoarea atributului *SQL%NOTFOUND* este *FALSE*.

Dacă o comandă *SELECT INTO* (nu este folosită clauza *BULK COLLECT*) întoarce mai multe linii, atunci apare imediat excepția *TOO_MANY_ROWS*.
În acest caz, atributul *SQL%ROWCOUNT* are valoarea 1 (nu numărul de linii care satisfac cererea).

5.2. Cursoare explicite

- Sunt cursoare la nivel de sesiune definite și gestionate de către utilizatori.
- Un cursor explicit are specificat un nume. Acesta este asociat cu o comandă *SELECT* ce întoarce de obicei mai multe linii.
- Mulțimea rezultat a cererii asociate poate fi procesată folosind una dintre variantele următoare:
  - se deschide cursorul (comanda *OPEN*), se încarcă liniile cursorului în variabile (comanda *FETCH*), se închide cursorul (comanda *CLOSE*);
  - se utilizează cursorul într-o comandă *FOR LOOP*.

5.2.1. Gestiunea cursoarelor explicite

```
DECLARE
   declarare cursor
BEGIN
   deschidere cursor (OPEN)
   WHILE rămân linii de recuperat LOOP
     recuperare linie rezultat (FETCH)
   END LOOP
   închidere cursor (CLOSE)
END;
```

Declararea unui cursor explicit

- Sintaxa de declarare, fără a asocia comanda *SELECT*

```
CURSOR nume_cursor [RETURN tip];
```

- Sintaxa de declarare, cu asocierea comenzii *SELECT*

```
CURSOR nume_cursor [RETURN tip]
IS comanda_SELECT;
```

Exemplul 5.2

```
DECLARE
  CURSOR c1 RETURN produse%ROWTYPE;

  CURSOR c2 IS
    SELECT id_produs, denumire FROM produse;

  CURSOR c1 RETURN produse%ROWTYPE IS
  SELECT * FROM PRODUSE;

BEGIN
  NULL;
END;
```

- Numele cursorului este un identificator unic în cadrul blocului, care nu poate să apară într-o expresie şi căruia nu i se poate atribui o valoare.
- Comanda *SELECT* care apare în declararea cursorului, nu trebuie să includă clauza *INTO*.
- Dacă se cere procesarea liniilor într-o anumită ordine, atunci în cerere este utilizată clauza *ORDER BY*.

- ❖ Variabilele care sunt referite în comanda *SELECT* trebuie declarate înaintea comenzii *CURSOR*. Acestea sunt considerate variabile de legătură.
- ❖ Dacă în lista *SELECT* apare o expresie, atunci pentru expresia respectivă trebuie utilizat un alias, iar câmpul expresie se va referi prin acest alias.

Deschiderea unui cursor explicit

- Sintaxa

```
OPEN nume_cursor;
```

- Comanda *OPEN* execută cererea asociată cursorului, identifică mulţimea liniilor rezultat (mulțimea activă) şi poziţionează cursorul înaintea primei linii.
- Dacă se încearcă deschiderea unui cursor deja deschis, atunci pare excepția *CURSOR_ALREADY_OPEN*.
- La deschiderea unui cursor se realizează următoarele operaţii:
  - se alocă resursele necesare pentru a procesa cererea
  - se procesează cererea
    - se evaluează comanda *SELECT* asociată (sunt examinate valorile variabilelor de legătură ce apar în declaraţia cursorului)
    - se identifică mulțimea activă prin execuția cererii *SELECT*, având în vedere valorile de la pasul anterior;
    - dacă cererea include clauza *FOR UPDATE*, atunci liniile din mulțimea activă sunt blocate;
  - se poziționează *pointer*-ul înaintea primei linii din mulţimea activă.

Exemplul 5.3

```
DECLARE

  CURSOR c1 IS
    SELECT * FROM categorii WHERE id_parinte IS NULL;

  CURSOR c2 IS
    SELECT * FROM categorii WHERE 1=2;


BEGIN
  OPEN c1;
  IF c1%FOUND THEN
     DBMS_OUTPUT.PUT_LINE('c1 - cel putin o linie');
  ELSE
     DBMS_OUTPUT.PUT_LINE('c1 - nicio linie');
  END IF;
```

```
  OPEN c2;
  IF c2%NOTFOUND THEN
     DBMS_OUTPUT.PUT_LINE('c2 - nicio linie');
  ELSE
     DBMS_OUTPUT.PUT_LINE('c2 - cel putin o linie');
  END IF;

  CLOSE c1;
  CLOSE c2;
END;
```

Încărcarea datelor dintr-un cursor explicit

- Comanda *FETCH* regăseşte liniile rezultatului din mulţimea activă.
- Sintaxa

```
FETCH nume_cursor INTO {nume_variabilă
      [, nume_variabilă] … | nume_înregistrare};

FETCH nume_cursor BULK COLLECT INTO
       {nume_variabilă_colecţie
        [,nume_variabilă_colecţie]
```

- *FETCH* realizează următoarele operaţii:
  - avansează *pointer*-ul la următoarea linie în mulţimea activă (*pointer*-ul poate avea doar un sens de deplasare de la prima înregistrare spre ultima);
  - citeşte datele liniei curente în variabile *PL/SQL*;
  - dacă *pointer*-ul este poziţionat la sfârşitul mulţimii active, atunci se iese din bucla cursorului.

Comanda *FETCH INTO* regăsește la un moment dat o singură linie.
Comanda *FETCH BULK COLLECT INTO* încarcă la un moment mai multe linii în colecții.

Exemplul 5.4 – vezi curs

Exemplul 5.5 – vezi curs

- Atunci când un cursor încarcă o linie, acesta realizează o „schimbare de context” – controlul este preluat de motorul *SQL* care va obține datele. Motorul *SQL* plasează datele în memorie și are loc o altă „schimbare de context” – controlul este preluat înapoi de motorul *PL/SQL*. Procesul se repetă până când nu mai sunt date de încărcat. Schimbările de context sunt foarte rapide, dar numărul prea mare de astfel de operații poate implica performanță scăzută.
- Folosind metoda *BULK COLLECT* sunt obținute mai multe linii, implicând doar 2 schimbări de context.
- Începând cu *Oracle 10g*, un cursor poate determina ca *PL/SQL* să realizeze implicit operații *BULK COLLECT*, încărcând câte 100 linii la un moment dat, fără a mai fi necesară utilizarea colecțiilor. În acest caz, utilizarea colecțiilor se poate dovedi utilă, doar dacă sunt încărcate mai multe sute de linii.

Exemplul 5.6 – vezi curs

Exemplul 5.7 – vezi curs

Închiderea unui cursor explicit

- După ce a fost procesată mulţimea activă, cursorul trebuie închis.
  - *PL/SQL* este informat că programul a terminat folosirea cursorului şi resursele asociate acestuia pot fi eliberate:
    - spaţiul utilizat pentru memorarea mulţimii active;
    - spaţiul temporar folosit pentru determinarea mulţimii active.
- Sintaxa:

```
CLOSE nume_cursor;
```

- Pentru a reutiliza cursorul este suficient ca acesta să fie redeschis.
- Dacă se încearcă încărcarea datelor dintr-un cursor închis, atunci apare excepţia *INVALID_CURSOR*.

- Dacă un bloc *PL/SQL* să termină fără a închide un cursor utilizat, sistemul nu va returna o eroare sau un mesaj de avertizare.
- Se recomandă închiderea cursoarelor pentru a permite sistemului să elibereze resursele alocate.

- Valorile atributelor unui cursor explicit sunt prezentate în următorul tabel:

	OPEN		Primul FETCH		Următorul FETCH		Ultimul FETCH		CLOSE	
	Înainte	După	Înainte	După	Înainte	După	Înainte	După	Înainte	După
%ISOPEN	False	True	True	True	True	True	True	True	True	False
%FOUND	Eroare	Null	Null	True	True	True	True	False	False	Eroare
%NOTFOUND	Eroare	Null	Null	False	False	False	False	True	True	Eroare
%ROWCOUNT	Eroare	0	0	1	1	Depinde de date				Eroare

- După prima încărcare, dacă mulţimea rezultat este vidă, *%FOUND* va fi *FALSE*, *%NOTFOUND* va fi *TRUE*, iar *%ROWCOUNT* este 0.

Procesarea liniilor unui cursor explicit

- Se utilizează o comandă de ciclare (*LOOP*, *WHILE* sau *FOR*), prin care la fiecare iteraţie se va încărca o linie nouă.
- Pentru ieşirea din ciclu poate fi utilizată comanda *EXIT*.
- Utilizarea comenzii de ciclare *LOOP*
  - vezi exemplul 5.4
- Utilizarea comenzii de ciclare *WHILE*
  - vezi exemplul 5.5

❖ Dacă se utilizează una dintre comenzile de ciclare *LOOP* sau *WHILE*, atunci cursorul trebuie:

1. declarat
2. deschis
3. parcurs, încărcând câte o linie la fiecare iteraţie (trebuie să se asigure ieşirea din buclă atunci când nu mai sunt linii de procesat)
4. închis

- Utilizarea comenzii de ciclare *FOR*
  - Procesarea liniilor unui cursor explicit se poate realiza şi cu ajutorul unui ciclu *FOR* special, numit **ciclu cursor**.
    - În acest caz cursorul trebuie doar declarat, operaţiile de deschidere, încărcare şi închidere ale acestuia fiind implicite.

- Sintaxa:

```
FOR nume_înregistrare IN nume_cursor LOOP
     secvenţă_de_instrucţiuni;
END LOOP;
```

  - Variabila *nume_înregistrare* nu trebuie declarată.

Exemplul 5.8 – vezi curs

- Există ciclu cursoare speciale care în comanda *FOR* în loc să refere un cursor declarat, utilizează direct o subcerere (**ciclu cursor cu subcereri**).
  - În acest caz nu este necesară nici măcar declararea cursorului.

Exemplul 5.9 – vezi curs

5.2.2. Cursoare parametrizate

- Unei variabile de tip cursor îi corespunde o comandă *SELECT*, care nu poate fi modificată pe parcursul programului.
- Cursoarele parametrizate sunt cursoare ale căror comenzi *SELECT* depind de parametri ce pot fi modificaţi la momentul execuţiei.
  - Transmiterea de parametri unui cursor parametrizat se face în mod similar procedurilor stocate.

Declararea unui cursor parametrizat

- Sintaxa de declarare, fără a asocia comanda *SELECT*

```
CURSOR nume_cursor (declarare_parametru
                    [,declarare_parametru …])
    [RETURN tip];
```

- Sintaxa de declarare, cu asocierea comenzii *SELECT*

```
CURSOR nume_cursor (declarare_parametru
                    [,declarare_parametru …])
    [RETURN tip]
IS comanda_SELECT;
```

- *declarare_parametru* are sintaxa:

```
nume_parametru [IN] tip_date_scalar
          [ {:= | DEFAULT} expresie]
```

  - Parametrul unui cursor nu poate fi declarat *NOT NULL*.

Deschiderea unui cursor parametrizat

- Se realizează asemănător apelului unei funcţii, specificând lista parametrilor actuali ai cursorului.
  - Asocierea dintre parametrii formali şi cei actuali se face prin:
    - poziţie (parametrii actuali sunt separaţi prin virgulă, respectând ordinea parametrilor formali);
    - nume (parametrii actuali sunt aranjaţi într-o ordine arbitrară, dar cu o corespondenţă de forma *parametru formal => parametru actual*).
  - Dacă în definiţia cursorului, toţi parametrii au valori implicite (*DEFAULT*), cursorul poate fi deschis fără a specifica vreun parametru.
- În determinarea mulţimii active se vor folosi valorile actuale ale parametrilor.
- Sintaxa

```
OPEN nume_cursor
 [ (valoare_parametru [, valoare_parametru] …) ];
```

Procesarea liniilor unui cursor parametrizat

- Dacă pentru procesare sunt utilizate comenzile de ciclare *LOOP* sau *WHILE*, atunci nu apar modificări de sintaxă.
- Dacă este utilizat un ciclu cursor, atunci se va utiliza sintaxa:

```
   FOR nume_înregistrare IN nume_cursor
    [(valoare_parametru [, valoare_parametru] …)] LOOP
         secvenţă_de_instrucţiuni;
    END LOOP;
```

Închiderea unui cursor parametrizat

- Nu apar modificări de sintaxă.

`Exemplul 5.10` **– vezi curs**

5.2.3. Cursoare *SELECT FOR UPDATE*

- Dacă este necesară blocarea liniilor înainte ca acestea să fie şterse sau reactualizate, atunci blocarea se poate realiza cu ajutorul clauzei *FOR UPDATE* a comenzii *SELECT* din definiția cursorului.
  - Cursorul trebuie să fie deschis.

- Sintaxa

```
CURSOR nume_cursor IS
   comanda_select
FOR UPDATE [OF listă_coloane]
           [NOWAIT | WAIT n | SKIP LOCKED];
```

  - Identificatorul *listă_coloane* este o listă ce include câmpurile tabelului care vor fi modificate.

    - Coloanele incluse în această listă indică doar liniile cărui tabel vor fi blocate.
    - Dacă lista de coloane lipsește, atunci vor fi blocate liniile selectate din toate tabelele referite în cerere.
  - Implicit comanda așteaptă până când linia necesară devine disponibilă și apoi întoarce rezultatul cererii.
  - Pentru a modifica acest comportament se poate utiliza una dintre opțiunile:
    - *NOWAIT* – nu așteaptă deblocarea liniei și întoarce o eroare dacă liniile sunt deja blocate de altă sesiune;
    - *WAIT n* – așteaptă *n* secunde (*n* este de tip întreg) pentru deblocarea liniei, iar dacă linia nu este deblocată în acest interval, întoarce un mesaj de eroare.
    - *SKIP LOCKED* – se va încerca blocarea liniilor selectate de cerere, iar liniile care sunt deja blocate de o altă tranzacție vor fi sărite (opțiune utilizată de exemplu în *Oracle Streams Advanced Queuing*).

```
Exemplul 5.11 - vezi explicatii curs

--sesiune 1
SELECT * FROM produse
WHERE id_produs=10 FOR UPDATE;
  --commit;

--sesiune 2
SELECT * FROM curs_plsql.produse
WHERE id_produs=10
FOR UPDATE NOWAIT;

SELECT * FROM curs_plsql.produse
WHERE id_produs=1000
FOR UPDATE WAIT 10;
```

- În momentul deschiderii unui cursor *FOR UPDATE*, liniile din mulţimea activă, determinată de clauza *SELECT*, sunt blocate pentru operaţii de scriere (reactualizare sau ştergere). În felul acesta este realizată consistenţa la citire a sistemului.
- De exemplu, această situaţie este utilă atunci când se reactualizează o valoare a unei linii şi trebuie avută siguranţa că linia nu este schimbată de un alt utilizator înaintea reactualizării. Astfel, alte sesiuni nu pot schimba liniile din mulţimea activă până când tranzacţia nu este permanentizată sau anulată.

- Dacă un cursor este declarat folosind clauza *FOR UPDATE*, atunci comenzile *DELETE*/*UPDATE* corespunzătoare trebuie să conţină clauza *WHERE CURRENT OF nume_cursor*.
  - Clauza referă linia curentă care a fost găsită de cursor, permiţând ca reactualizările şi ştergerile să se efectueze asupra acestei linii, fără referirea explicită a cheii primare sau pseudocoloanei *ROWID*.

- Deoarece cursorul lucrează doar cu nişte copii ale liniilor existente în tabele, după închiderea cursorului este necesară comanda *COMMIT* pentru a realiza scrierea efectivă a modificărilor.
- Deoarece blocările implicate de clauza *FOR UPDATE* vor fi eliberate de comanda *COMMIT*, nu este recomandată utilizarea comenzii *COMMIT* în interiorul ciclului în care se fac încărcări de date. Orice *FETCH* executat după *COMMIT* va eşua.
- În cazul în care cursorul nu este definit folosind *SELECT…FOR UPDATE*, nu apar probleme în acest sens şi, prin urmare, în interiorul ciclului unde se fac schimbări ale datelor poate fi utilizată comanda *COMMIT*.

`Exemplul 5.12` – **vezi curs**

`Exemplul 5.13` – **vezi curs**

5.2.4. Cursoare dinamice

- Un cursor static este un cursor a cărui comandă *SQL* este cunoscută la momentul compilării blocului.
  - Toate exemplele anterioare se referă la cursoare statice.
- În *PL/SQL* a fost introdus conceptul de variabilă cursor, care este de tip referinţă.
- Variabilele cursor
  - sunt similare tipului *pointer* din limbajele *C* sau *Pascal*
    - un cursor este un obiect static, iar un cursor dinamic este un *pointer* la un cursor
  - sunt dinamice deoarece 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
  - la momentul declarării nu solicită o cerere asociată
  - pot primi valori
  - pot fi utilizate în expresii
  - pot fi utilizate ca parametrii în subprograme
    - pot fi utilizate pentru a transmite mulțimea rezultat a unei cereri între subprograme
  - pot fi variabile de legătură
    - pot fi utilizate pentru a transmite mulțimea rezultat a unei cereri între subprograme stocate și diferiți clienți
  - nu acceptă parametrii
- Sintaxa de declarare

```
TYPE  tip_ref_cursor IS REF CURSOR [RETURN tip_returnat];
var_cursor  tip_ref_cursor;
```

  - *var_cursor* este numele variabilei cursor
  - *tip_ref_cursor* este un nou tip de dată ce poate fi utilizat în declaraţiile următoare ale variabilelor cursor
  - *tip_returnat* este un tip înregistrare sau tipul unei linii dintr-un tabel
    - corespunde coloanelor întoarse de către orice cursor asociat variabilelor cursor de tipul definit

        - dacă lipseşte clauza *RETURN*, cursorul poate fi deschis pentru orice cerere
- Restricţii de utilizare a variabilelor cursor
    - variabilele cursor nu pot fi declarate în specificația unui pachet
        - un pachet nu poate avea definită o variabilă cursor ce poate fi referită din afara pachetului
    - valoarea unei variabile cursor nu poate fi stocată într-o colecție sau o coloană a unui tabel
    - nu pot fi utilizaţi operatorii de comparare pentru a testa egalitatea, inegalitatea sau valoarea *null* a variabilelor cursor
    - nu pot fi folosite cu *SQL dinamic* în *Pro*C/C++*

Utilizarea unei variabile cursor

- Comanda *OPEN...FOR* asociază o variabilă cursor cu o cerere, execută cererea, identifică mulţimea rezultat şi poziţionează cursorul înaintea primei linii din mulţimea rezultat.
- Sintaxa

```
OPEN {variabila_cursor | :variabila_cursor_host}
FOR {cerere_select |
    şir_dinamic [USING argument_bind [, argument_bind …]]};
```

    - *variabila_cursor* specifică o variabilă cursor declarată anterior
    - *cerere_select* reprezintă cererea pentru care este deschisă variabila cursor
    - *şir_dinamic* este o secvenţă de caractere care reprezintă cererea
        - este specifică prelucrării dinamice a comenzilor, iar posibilităţile oferite de *SQL* dinamic vor fi analizate într-un capitol separat
    - *:variabila_cursor_host* reprezintă o variabilă cursor declarată într-un mediu gazdă *PL/SQL*
- Comanda *OPEN .. FOR* poate deschide acelaşi cursor pentru diferite cereri. Nu este necesară închiderea variabilei cursor înainte de a o redeschide. Dacă se redeschide variabila cursor pentru o nouă cerere, cererea anterioară este pierdută.

  `Exemplul 5.14` – **vezi curs**

  `Exemplul 5.15` – **vezi curs**

Expresii cursor

- În versiunea *Oracle9i* a fost introdus conceptul de expresie cursor, care întoarce un cursor imbricat (*nested cursor*).
- Sintaxa:

```
CURSOR (subcerere)
```

- Semnificație
  - Fiecare linie din mulţimea rezultat poate conţine valori uzuale şi cursoare generate de subcereri.
- Utilizare
  - *PL/SQL* acceptă cereri care au expresii cursor în cadrul unei declaraţii cursor, declaraţii *REF CURSOR* şi a variabilelor cursor.
    - Expresia cursor poate să apară într-o comandă *SELECT* ce este utilizată pentru deschiderea unui cursor dinamic.
    - Expresia cursor poate fi utilizată în cereri SQL dinamice sau ca parametri actuali într-un subprogram.
  - Restricții de utilizare a unei expresii cursor
    - nu poate fi utilizată cu un cursor implicit;
    - poate să apară numai într-o comandă *SELECT* care nu este imbricată în altă cerere (exceptând cazul în care este o subcerere chiar a expresiei cursor) sau ca argument pentru funcţii tabel, în clauza *FROM* a lui *SELECT*;
    - nu poate să apară în interogarea ce defineşte o vizualizare;
    - nu se pot efectua operaţii *BIND* sau *EXECUTE* cu aceste expresii.
- Încărcarea cursorului imbricat se realizează
  - automat atunci când liniile care îl conţin sunt încărcate din cursorul „părinte“.
- Închiderea cursorului imbricat are loc
  - dacă este realizată explicit de către utilizator;
  - atunci când cursorul „părinte“ este reexecutat sau închis;
  - dacă apare o eroare în timpul unei încărcări din cursorul „părinte“.

Exemplul 5.16 – vezi curs

Bibliografie

1. Connolly T.M., Begg C.E., Database Systems: *A Practical Approach to Design, Implementation and Management*, 5th edition, Pearson Education, 2005
2. Dollinger R., Andron L., *Baze de date şi gestiunea tranzacţiilor*, Editura Albastră, Cluj-Napoca, 2004
3. Oracle and/or its affiliates, *Oracle Database Concepts*, 1993, 2017
4. Oracle and/or its affiliates, *Oracle Database Performance Tuning Guide*, 2013, 2017
5. Oracle and/or its affiliates, *Oracle Database SQL Language Reference*, 1996, 2017
6. Oracle and/or its affiliates, *Oracle Database PL/SQL Language Reference*, 1996, 2017
7. Oracle and/or its affiliates, *Oracle Database Administrator's Guide*, 2001, 2010
8. Oracle and/or its affiliates, *Pro*C/C++ Programmer's Guide*, 1996, 2014
9. Oracle University, *Oracle Database 11g*: *PL/SQL Fundamentals*, *Student Guide*, 2009
10. Popescu I., Alecu A., Velcescu L., Florea (Mihai) G., *Programare avansată în Oracle9i*, Ed. Tehnică, 2004