Cursoare

Gestiunea cursoarelor in *PL/SQL*

Sistemul *Oracle* folose§te, pentrn a procesa o comanda *SQL, o* zona de memorie cunoscuta sub numele de zona context *(context area).* Cand este procesata o instructiune *SQL,* server-ul *Oracle* deschide aceasta zona de memorie in care comanda este analizata sintactic §i este executata.

Zona confine informafii necesare procesarii comenzii, cum ar fi:

- numarul de randuri procesate de instrucfiune;
- un*pointer* catre reprezentarea interna a comenzii;
- in cazul unei cereri, mulfimea randurilor rezultate in urma execufiei acestei comenzi *(active set).*

Un cursor este un *pointer* la aceasta zona context. Prin intermediul cursoarelor, un program *PL/SQL* poate controla zona context §i transformarile petrecute in urma procesarii comenzii.

Exista doua tipuri de cursoare:

- implicite, generate de server-ul *Oracle* cand in partea executabila a unui bloc *PL/SQL* apare o instructiune *SQL*;
- explicite, declarate §i definite de catre utilizator atunci cand o cerere *(SELECT),* care apare intr-un bloc *PL/SQL,* intoarce mai multe linii ca rezultat.

Atat cursoarele implicite cat §i cele explicite au o serie de atribute ale caror valori pot fi folosite in expresii. Lista atributelor este urmatoarea:

- *%ROWCOUNT,* care este de tip intreg §i reprezinta numarul liniilor incarcate de cursor;
- *%FOUND,* care este de tip boolean §i ia valoarea *TRUE* daca ultima operafie de incarcare *(FETCH)* dintr-un cursor a avut succes (in cazul cursoarelor explicite) sau daca instrucfiunea *SQL* a intors cel pufin o linie (in cazul cursoarelor implicite);
- *%NOTFOUND,* care este de tip boolean §i are semnificafie opusa fata de cea a atributului *%FOUND*;
- *%ISOPEN,* care este de tip boolean §i indica daca un cursor este deschis (in cazul cursoarelor implicite, acest atribut are intotdeauna valoarea *FALSE,* deoarece un cursor implicit este inchis de sistem imediat dupa executarea instrucfiunii *SQL* asociate).

Atributele pot fi referite prin expresia *SQL%nume_atribut,* in cazul cursoarelor implicite, sau prin *nume_cursor%nume_atribut,* in cazul unui cursor explicit. Ele pot sa apara in comenzi *PL/SQL,* in funcfii, in secfiunea de tratare a erorilor, dar nu pot fi utilizate in comenzi *SQL.*

Cursoare implicite

Cand se proceseaza o comanda *LMD*, motorul *SQL* deschide un cursor implicit. Atributele scalare ale cursorului implicit *(SQL%ROWCOUNT, SQL%FOUND, SQL%NOTFOUND, SQL%ISOPEN)* furnizeaza informal referitoare la ultima comanda *INSERT, UPDATE, DELETE* sau *SELECT INTO* executata. inainte ca *Oracle* sa deschida cursorul *SQL* implicit, atributele acestuia au valoarea *null.*

in *Oracle,* pentru cursoare implicite a fost introdus atributul compus *%BULK_ROWCOUNT,* care este asociat comenzii *FORALL.* Atributul are semantica unui tablou indexat. Componenta *%BULK_ROWCOUNT(j)* confine numarul de linii procesate de a j-a execute a unei comenzi *INSERT, DELETE* sau *UPDATE.* Daca a j-a execute nu afecteaza nici o linie, atunci atributul returneaza valoarea 0. Comanda *FORALL* §i atributul *%BULK_ROWCOUNT* au aceia§i indici, deci folosesc acela§i domeniu. Daca *%BULK_ROWCOUNT(j)* este zero, atributul *%FOUND* este *FALSE.*

Exemplu:

in exemplul care urmeaza, comanda *FORALL* insereaza un numar arbitrar de linii la fiecare iterate, iar dupa fiecare iterate atributul *%BULK_ROWCOUNT* returneaza numarul acestor linii inserate.

```
SET SERVEROUTPUT ON DECLARE
TYPE alfa IS TABLE OF NUMBER; beta alfa;
BEGIN
SELECT cod_artist BULK COLLECT INTO beta FROM artist;
FORALL j IN 1..beta.COUNT INSERT INTO tab_art
SELECT cod_artist,cod_opera FROM opera
WHERE cod_artist = beta(j);
FOR j IN 1..beta.COUNT LOOP
DBMS_OUTPUT.PUT_LINE (’Pentru artistul avand codul ’
||
beta(j) ||    ’ au fost inserate ’    ||
SQL%BULK_ROWCOUNT(j)
|| inregistrari (opere de arta)’);
END LOOP;
DBMS_OUTPUT.PUT_LINE (’Numarul total de inregistrari
Inserate este ’||SQL%ROWCOUNT);
END;
/
SET SERVEROUTPUT OFF
```

Cursoare explicite

Pentrn gestiunea cursoarelor explicite sunt necesare urmatoarele etape:

- declararea cursorului (atribuirea unui nume §i asocierea cu o comanda *SELECT);*
- deschiderea cursorului pentru cerere (executarea interogarii asociate §i determinarea multimii rezultat);
- recuperarea liniilor rezultatului in variabile *PL/SQL;*
- inchiderea cursorului (eliberarea resurselor relative la cursor).

Prin urmare, pentru a utiliza un cursor, el trebuie declarat in sec|iunea declarativa a programului, trebuie deschis in partea executabila, urmand sa fie utilizat apoi pentru extragerea datelor. Daca nu mai este necesar in restul programului, cursorul trebuie sa fie inchis.

DECLARE
declarare cursor BEGIN
deschidere cursor (OPEN)
WHILE raman linii de recuperat LOOP recuperare linie rezultat (FETCH)

END LOOP
inchidere cursor (CLOSE)

END;

Pentru a controla activitatea unui cursor sunt utilizate comenzile *DECLARE, OPEN,, FETCH* §i *CLOSE.*

Declararea unui cursor explicit

Prin declaratia *CURSOR* in cadrul comenzii *DECLARE* este definit un cursor explicit §i este precizata structura cererii care va fi asociata acestuia. Declaratia *CURSOR* are urmatoarea forma sintactica:

CURSOR nume cursor IS comanda select

Identificatorul *nume cursor* este numele cursorului, iar *comanda select* este cererea *SELECT* care va fi procesata.

Observafii:

- Comanda *SELECT* care apare in declararea cursorului, nu trebuie sa includa clauza *INTO.*

- Daca se cere procesarea liniilor intr-o anumita ordine, atunci in cerere este utilizata clauza *ORDER BY.*
- Variabilele care sunt referite in comanda de selectare trebuie declarate inaintea comenzii *CURSOR.* Ele sunt considerate variabile de legatura.
- Daca in lista comenzii *SELECT* apare o expresie, atunci pentru expresia respectiva trebuie utilizat un *alias*, iar campul expresie se va referi prin acest *alias.*
- Numele cursorului este un identificator unic in cadrul blocului, care nu poate sa apara intr-o expresie §i caruia nu i se poate atribui o valoare.

Deschiderea unui cursor explicit

Comanda *OPEN* executa cererea asociata cursorului, identifica multimea liniilor rezultat §i pozitioneaza cursorul inaintea primei linii.

Deschiderea unui cursor se face prin comanda:

OPEN numecursor;

Identificatorul *nume cursor* reprezinta numele cursorului ce va fi deschis.

La deschiderea unui cursor se realizeaza urmatoarele operatii:

- se evalueaza cererea asociata (sunt examinate valorile variabilelor de legatura ce apar in declaratia cursorului);
- este determinata multimea rezultat *(active set)* prin executarea cererii *SELECT*, avand in vedere valorile de la pasul anterior;
- *pointer-ul* este pozitionat la prima linie din multimea activa.

Incarcarea datelor dintr-un cursor explicit

Comanda *FETCH* regase§te liniile rezultatului din multimea activa. *FETCH* realizeaza urmatoarele operatii:

- avanseaza *pointer-ul* la urmatoarea linie in multimea activa (*pointer-ul* poate avea doar un sens de deplasare de la prima spre ultima inregistrare);
- cite§te datele liniei curente in variabile *PL/SQL*;
- daca *pointer-ul* este pozitionat la sfar§itul multimii active atunci se iese din bucla cursorului.

Comanda *FETCH* are urmatoarea sintaxa:

FETCH nume_cursor

INTO {numejvariabila [, *nume variabila] ...* | *nume inregistrare);*

Identificatorul *nume cursor* reprezinta numele unui cursor declarat §i deschis anterior. Variabila sau lista de variabile din clauza *INTO* trebuie sa fie

compatibila (ca ordine §i tip) cu lista selectata din cererea asociata cursorului.
La un moment dat, comanda *FETCH* regase§te o singura linie. Totu§i, in ultimele versiuni *Oracle* pot fi incarnate mai multe linii (la un moment dat) intro colectie, utilizand clauza *BULK COLLECT.*

Exemplu:
in exemplul care urmeaza se incarca date dintr-un cursor in doua colectii.

```
DECLARE
TYPE    ccopera IS TABLE OF opera.cod_opera%TYPE;
TYPE    ctopera IS TABLE OF opera.titlu%TYPE;
cod1 ccopera; titlu1 ctopera;
CURSOR alfa IS SELECT cod_opera, titlu
FROM opera
WHERE stil = 'impresionism';
BEGIN
OPEN alfa;
FETCH alfa BULK COLLECT INTO cod1, titlu1;

CLOSE alfa;
END;
```

Inchiderea unui cursor explicit

Dupa ce a fost procesata multimea activa, cursorul trebuie inchis. Prin aceasta operate, *PL/SQL* este informat ca programul a terminat folosirea cursorului §i resursele asociate acestuia pot fi eliberate. Aceste resurse includ spatiul utilizat pentru memorarea multimii active §i spatiul temporar folosit pentru determinarea multimii active.
Cursorul va fi inchis prin comanda *CLOSE,* care are urmatoarea sintaxa:

***CLOSE** numecursor;*

Identificatorul *nume cursor* este numele unui cursor deschis anterior. Pentru a reutiliza cursorul este suficient ca acesta sa fie redeschis. Daca se incearca incarcarea datelor dintr-un cursor inchis, atunci apare exceptia *INVALID CURSOR.* Un bloc *PL/SQL* poate sa se termine fara a inchide cursoarele, dar acest lucru nu este indicat, deoarece este bine ca resursele sa fie eliberate.

Exemplu:
Pentru toti arti§tii care au opere de arta expuse in muzeu sa se insereze in tabelul *temp* informatii referitoare la numele acestora §i anul na§terii.

```
DECLARE
v_nume    artist.nume%TYPE;
v_an_nas    artist.an_nastere%TYPE;
```

```
  CURSOR info IS
    SELECT DISTINCT nume, an_nastere FROM artist;
BEGIN
  OPEN info;
  LOOP
    FETCH info INTO v_nume, v_an_nas;
    EXIT WHEN info%NOTFOUND;
    INSERT INTO temp
    VALUES (v_nume || TO_CHAR(v_an_nas));
  END LOOP;
  CLOSE info;
  COMMIT;
END;
```

Valorile atributelor unui cursor explicit sunt prezentate in urmatorul tabel:

		%FOUND	***%ISOPEN***	***%NOTFOUND***	***%ROWCOUNT***
OPEN	**inainte**	**Exceplie**	*False*	**Exceplie**	**Exceplie**
	Dupa	*Null*	*True*	*Null*	**0**
Prima	**inainte**	*Null*	*True*	*Null*	**0**
Incarcare	**Dupa**	*True*	*True*	*False*	**1**
Urmatoarea	**inainte**	*True*	*True*	*False*	**1**
incarcare	**Dupa**	*True*	*True*	*False*	**Depinde de date**
Ultima	**inainte**	*True*	*True*	*False*	**Depinde de date**
incarcare	**Dupa**	*False*	*True*	*True*	**Depinde de date**
CLOSE	**inainte**	*False*	*True*	*True*	**Depinde de date**
	Dupa	**Exceplie**	*False*	**Exceplie**	**Exceplie**

Dupa prima incarcare, daca mullimea rezultat este vida, *%FOUND* va fi *FALSE*, *%NOTFOUND* va fi *TRUE*, iar *%ROWCOUNT* este 0.

intr-un pachet poate fi separata specificarea unui cursor de corpul acestuia. Cursorul va fi declarat in specificalia pachetului prin comanda:

CURSOR nume cursor [*(parametru* [*, parametru*]...)]

RETURN tip_ returnat;

in felul acesta va create flexibilitatea programului, putand fi modificat doar corpul cursorului, fara a schimba specificalia.

Exemplu:

```
CREATE PACKAGE exemplu AS
CURSOR alfa    (p_valoare_min    NUMBER) RETURN
opera%ROWTYPE;

-- declaratie specificatie cursor END exemplu;
CREATE PACKAGE BODY exemplu AS
CURSOR alfa (p_valoare_min NUMBER) RETURN opera%ROWTYPE IS
SELECT * FROM opera WHERE valoare > p_valoare_min;
-- definire corp cursor

END exemplu;
```

Procesarea liniilor unui cursor explicit

Pentru procesarea diferitelor linii ale unui cursor explicit se folose§te operalia de ciclare *(LOOP, WHILE, FOR),* prin care la fiecare iterate se va incarca o noua linie. Comanda *EXIT* poate fi utilizata pentru ie§irea din ciclu, iar valoarea atributului *%ROWCOUNT* pentru terminarea ciclului.

Procesarea liniilor unui cursor explicit se poate realiza §i cu ajutorul unui ciclu *FOR* special, numit ciclu cursor. Pentru acest ciclu este necesara doar declararea cursorului, operable de deschidere, incarcare §i inchidere ale acestuia fiind implicite.

Comanda are urmatoarea sintaxa:

FOR numeinregistrare IN nume cursor LOOP secvenfddejnstrucfiuni;
END LOOP;

Variabila *numeinregistrare* (care controleaza ciclul) nu trebuie declarata. Domeniul ei este doar ciclul respectiv.

Pot fi utilizate cicluri cursor speciale care folosesc subcereri, iar in acest caz nu mai este necesara nici declararea cursorului. Exemplul care urmeaza este concludent in acest sens.

Exemplu:

Sa se calculeze, utilizand un ciclu cursor cu subcereri, valoarea operelor de arta expuse intr-o galerie al carei cod este introdus de la tastatura. De asemenea, sa se oblina media valorilor operelor de arta expuse in galeria respectiva.

```
SET SERVEROUTPUT ON
ACCEPT p_galerie PROMPT 'Dati codul galeriei:'
DECLARE
v_cod_galerie galerie.cod_galerie%TYPE:=&p_galerie;
```

```
  val     NUMBER
  media   ;
i    INTEGER;
BEGIN    ;
val:=0;
i:=0;
FOR numar_opera IN
(SELECT cod_opera, valoare FROM    opera
WHERE    cod_galerie = v_cod_galerie) LOOP
val := val + numar_opera.valoare; i := i+1;
END LOOP;--inchidere implicita
DBMS_OUTPUT.PUT_LINE('Valoarea operelor de arta din
galeria cu numarul ’    || TO_CHAR(v_cod_galerie) ||    ’
     este ’    || TO_CHAR(val));
IF i=0 THEN
DBMS_OUTPUT.PUT_LINE('Galeria nu are opere de arta’); ELSE
     media := val/i;
     DBMS_OUTPUT.PUT_LINE('Media valorilor operelor de
arta
din galeria cu numarul ’    || TO_CHAR(v_cod_galerie)
||    ’ este ’    || TO_CHAR(media));
END IF;
END;
/
SET SERVEROUTPUT OFF
```

Cursoare parametrizate

Unei variabile de tip cursor ii corespunde o comanda *SELECT*, care nu poate fi schimbata pe parcursul programului. Pentru a putea lucra cu ni§te cursoare ale caror comenzi *SELECT* ata§ate depind de parametri ce pot fi modificati la momentul executiei, in *PL/SQL* s-a introdus notiunea de cursor parametrizat. Prin urmare, un cursor parametrizat este un cursor in care comanda *SELECT* ata§ata depinde de unul sau mai multi parametri.

Transmiterea de parametri unui cursor parametrizat se face in mod similar procedurilor stocate. Un astfel de cursor este mult mai u§or de interpretat §i de intretinut, oferind §i posibilitatea reutilizarii sale in blocul *PL/SQL*.

Declararea unui astfel de cursor se face respectand urmatoarea sintaxa:

CURSOR nume cursor [*(nume_parametru[, nume_parametru ...*])] *[RETURN tipreturnat]*
IS comandaselect;

Identificatorul *comanda select* este o instructive *SELECT* fara clauza *INTO*, *tipreturnat* reprezinta un tip inregistrare sau linie de tabel, iar *nume_parametru* are sintaxa:

nume_parametru [IN] *tip_parametru* [{:= | *DEFAULT} expresie]*

in aceasta declaratie, atributul *tip_parametru* reprezinta tipul parametrului, care este un tip scalar. Parametrii formali sunt de tip *IN* §i, prin urmare, nu pot returna valori parametrilor actuali. Ei nu suporta constrangerea
NOT NULL.

Deschiderea unui astfel de cursor se face asemanator apelului unei functii, specificand lista parametrilor actuali ai cursorului. in determinarea multimii active se vor folosi valorile actuale ale acestor parametri.

Sintaxa pentru deschiderea unui cursor parametrizat este:

OPEN numecursor [*(valoare_parametru [, valoare_parametru] ...*)];

Parametrii sunt specificati similar celor de la subprograme. Asocierea dintre parametrii formali §i cei actuali se face prin:

- pozitie - parametrii formali §i actuali sunt separati prin virgula;
- nume - parametrii actuali sunt aranjati intr-o ordine arbitrara, dar cu o corespondenta de forma *parametru formal => parametru actual.*

Daca in definitia cursorului, toti parametrii au valori implicite *(DEFA ULT),* cursorul poate fi deschis fara a specifica vreun parametru.

Exemplu:

Utilizand un cursor parametrizat sa se obtina codurile operelor de arta din fiecare sala, identificatorul salii §i al galeriei. Rezultatele sa fie inserate in tabelul *mesaje.*

```
DECLARE
v_cod_sala    sala.cod_sala%TYPE;
v_cod_galerie galerie.cod_galerie%TYPE; v‟car~
VARCHAR2(75);
CURSOR sala_cursor IS
SELECT    cod_sala,cod_galerie
FROM    sala;
CURSOR ope_cursor (v_id_sala NUMBER,v_id_galerie NUMBER) IS
SELECT cod_opera || cod_sala || cod_galerie FROM    opera
WHERE    cod_sala = v_id_sala
AND    cod_galerie = v_id_galerie;
BEGIN    -  -  -
OPEN sala_cursor;
LOOP
FETCH sala_cursor INTO v_cod_sala,v_cod_galerie;
```

```
EXIT WHEN sala_cursor%NOTFOUND;
IF ope_cursor%ISOPEN THEN CLOSE ope_cursor;
END IF;
OPEN ope_cursor (v_cod_sala, v_cod_galerie);
LOOP
FETCH ope_cursor INTO v_car;
EXIT WHEN ope_cursor%NOTFOUND;
INSERT INTO mesaje (rezultat)
VALUES (v_car);
END LOOP;
CLOSE ope_cursor;
END LOOP;
CLOSE sala_cursor;
COMMIT;
END;
```

Cursoare *SELECT FOR UPDATE*

Uneori este necesara blocarea liniilor inainte ca acestea sa fie §terse sau reactualizate. Blocarea se poate realiza (atunci cand cursorul este deschis) cu ajutorul comenzii *SELECT* care confine clauza *FOR UPDATE*.

Declararea unui astfel de cursor se face conform sintaxei:

***CURSOR** numecursor **IS** comandaselect*
***FOR UPDATE [OF** listacampuri] [**NOWAIT**];*

Identificatorul *lista campuri* este o lista ce include campurile tabelului care vor fi modificate. Atributul *NOWAIT* returneaza o eroare daca liniile sunt deja blocate de alta sesiune. Liniile unui tabel sunt blocate doar daca clauza *FOR UPDATE* se refera la coloane ale tabelului respectiv.

in momentul deschiderii unui astfel de cursor, liniile corespunzatoare mullimii active, determinate de clauza *SELECT,* sunt blocate pentru operalii de scriere (reactualizare sau §tergere). in felul acesta este realizata consistent la citire a sistemului. De exemplu, aceasta situate este utila cand se reactualizeaza o valoare a unei linii §i trebuie avuta siguranla ca linia nu este schimbata de alt utilizator lnaintea reactualizarii. Prin urmare, alte sesiuni nu pot schimba liniile din mullimea activa pana cand tranzaclia nu este permanentizata sau anulata. Daca alta sesiune a blocat deja liniile din mullimea activa, atunci comanda *SELECT ... FOR UPDATE* va a§tepta (sau nu) ca aceste blocari sa fie eliberate. Pentru a trata aceasta situalie se utilizeaza clauza *WAIT,* respectiv *NOWAIT.*

in *Oracle* este utilizata sintaxa:

***SELECT** … **FROM** … **FOR UPDATE [OF**lista campuri]*
[*{**WAIT**n* | ***NOWAIT**}*];

Valoarea lui *n* reprezinta numarul de secunde de a§teptare. Daca liniile nu sunt deblocate in *n* secunde, atunci se declan§eaza eroarea *ORA-30006*, respectiv eroarea *ORA-00054*, dupa cum este specificata clauza *WAIT*, respectiv *NOWAIT*. Daca nu este specificata nici una din clauzele *WAIT* sau *NOWAIT*, sistemul a§teapta pana ce linia este deblocata §i atunci returneaza rezultatul comenzii *SELECT*.

Daca un cursor este declarat cu clauza *FOR UPDATE*, atunci comenzile *DELETE* §i *UPDATE* corespunzatoare trebuie sa confina clauza *WHERE CURRENT OF numecursor*.

Aceasta clauza refera linia curenta care a fost gasita de cursor, permijand ca reactualizarile §i §tergerile sa se efectueze asupra acestei linii, fara referirea explicita a cheii primare sau pseudocoloanei *ROWID*. De subliniat ca instrucjiunile *UPDATE* §i *DELETE* vor reactualiza numai coloanele listate in clauza *FOR UPDATE*.

Pseudocoloana *ROWID* poate fi utilizata daca tabelul referit in interogare nu are o cheie primara specificata. *ROWID-ul* fiecarei linii poate fi incarcat intro variabila *PL/SQL* (declarata de tipul *ROWID* sau *UROWID)*, iar aceasta variabila poate fi utilizata in clauza *WHERE (WHERE ROWID = v rowid)*.

Dupa inchiderea cursorului este necesara comanda *COMMIT* pentru a realiza scrierea efectiva a modificarilor, deoarece cursorul lucreaza doar cu ni§te copii ale liniilor reale existente in tabele.

Deoarece blocarile implicate de clauza *FOR UPDATE* vor fi eliberate de comanda *COMMIT*, nu este recomandata utilizarea comenzii *COMMIT* in interiorul ciclului in care se fac incarcari de date. Orice *FETCH* executat dupa *COMMIT* va e§ua. in cazul in care cursorul nu este definit prin *SELECT...FOR UPDATE*, nu sunt probleme in acest sens §i, prin urmare, in interiorul ciclului unde se fac schimbari ale datelor poate fi utilizat un *COMMIT*.

Exemplu:

Sa se dubleze valoarea operelor de arta pictate pe panza care au fost achizijionate inainte de 1 ianuarie 1956.

```
DECLARE
CURSOR calc IS SELECT *
FROM opera
WHERE material = 'panza'
AND    data_achizitie <= TO_DATE('01-JAN-56','DD-MON-
YY')
FOR UPDATE OF valoare NOWAIT;
BEGIN
FOR x IN calc LOOP UPDATE opera
```

```
    SET    valoare = valoare*2
    WHERE CURRENT OF calc;
  END LOOP;
  -- se permanentizeaza actiunea si se elibereaza blocarea
  COMMIT;
END;
```

Cursoare dinamice

Toate exemplele considerate anterior se refera la cursoare statice. Unui cursor static i se asociaza o comanda *SQL* care este cunoscuta in momentul in care blocul este compilat.

in *PL/SQL* a fost introdusa variabila cursor, care este de tip referinja. Variabilele cursor sunt similare tipului *pointer* din limbajele *C* sau *Pascal***.** Prin urmare, un cursor este un obiect static, iar un cursor dinamic este un *pointer* la un cursor.

in momentul declararii, variabilele cursor nu solicita o comanda *SQL* asociata. in acest fel, diferite comenzi *SQL* pot fi asociate variabilelor cursor, la diferite momente de timp. Acest tip de variabila trebuie declarata, deschisa, incarcata §i inchisa in mod similar unui cursor static.

Variabilele cursor sunt dinamice deoarece li se pot asocia diferite interogari atata timp cat coloanele returnate de fiecare interogare corespund declarajiei variabilei cursor.

Aceste variabile sunt utile in transmiterea seturilor de rezultate intre subprograme *PL/SQL* stocate §i diferiji clienji. De exemplu, un *client OCI***,** o aplicajie *Oracle Forms* §i server-ul *Oracle* pot referi aceea§i zona de lucru (care conjine muljimea rezultat). Pentru a reduce traficul in retea, o variabila cursor poate fi declarata pe stajia *client***,** deschisa §i se pot incarca date din ea pe *server***,** apoi poate continua incarcarea, dar de pe stajia *client* etc.

Pentru a crea o variabila cursor este necesara definirea unui tip *REF CURSOR*, urmand apoi declararea unei variabile de tipul respectiv. Dupa ce variabila cursor a fost declarata, ea poate fi deschisa pentru orice cerere *SQL* care returneaza date de tipul declarat.

Sintaxa pentru declararea variabilei cursor este urmatoarea:

TYPE** tip ref cursor **IS REF CURSOR [RETURN** tip returnat**];

var cursor tip ref cursor;

Identificatorul *var cursor* este numele variabilei cursor, *tip ref cursor* este un nou tip de data ce poate fi utilizat in declarable urmatoare ale variabilelor cursor, iar *tip returnat* este un tip inregistrare sau tipul unei linii dintr-un tabel al bazei. Acest tip corespunde coloanelor returnate de catre orice cursor asociat variabilelor cursor de tipul definit. Daca lipse§te clauza *RETURN***,**

cursorul poate fi deschis pentru orice cerere *SELECT.*
Daca variabila cursor apare ca parametru intr-un subprogram, atunci trebuie specificat tipul parametrului (tipul *REF CURSOR)* §i forma acestuia *(IN* sau *IN OUT).*

Exista anumite restricfii referitoare la utilizarea variabilelor cursor:

- nu pot fi declarate intr-un pachet;
- nu poate fi asignata valoarea *null* unei variabile cursor;
- nu poate fi utilizat tipul *REF CURSOR* pentru a specifica tipul unei coloane in comanda *CREATE TABLE;*
- nu pot fi utilizafi operatorii de comparare pentru a te sta egalitatea, inegalitatea sau valoarea *null* a variabilelor cursor;
- nu poate fi utilizat tipul *REF CURSOR* pentru a specifica tipul elementelor unei colecfii *(varray, nested table);*
- nu pot fi folosite cu *SQL* dinamic in *Pro *C/C++*.

in cazul variabilelor cursor, instrucfiunile de deschidere *(OPEN),* incarcare *(FETCH),* inchidere *(CLOSE)* vor avea o sintaxa similara celor comentate anterior.

Comanda *OPEN...FOR* asociaza o variabila cursor cu o cerere multilinie, executa cererea, identifica mulfimea rezultat §i pozifioneaza cursorul la prima linie din mulfimea rezultat. Sintaxa comenzii este:

OPEN {variabilacursor | wariabilacursorhost}
FOR { *cerere_select |*
§ir_dinamic [USING argumentbind [, argument_bind ...]] };

Identificatorul *variabila cursor* specifica o variabila cursor declarata anterior, dar fara opfiunea *RETURN tip, cerere select* este interogarea pentru care este deschisa variabila cursor, iar *§ir_dinamic* este o secvenfa de caractere care reprezinta cererea multilinie.
Opfiunea *§ir_dinamic* este specifica prelucrarii dinamice a comenzilor, iar posibilitafile oferite de *SQL* dinamic vor fi analizate intr-un capitol separat. Identificatorul *wariabila cursor host* reprezinta o variabila cursor declarata intr-un mediu gazda *PL/SQL* (de exemplu, un program *OCI).*

Comanda *OPEN - FOR* poate deschide acela§i cursor pentru diferite cereri. Nu este necesara inchiderea variabilei cursor inainte de a o redeschide. Daca se redeschide variabila cursor pentru o noua cerere, cererea anterioara este pierduta.

Exemplu:

```
CREATE OR REPLACE PACKAGE alfa AS
TYPE ope_tip IS REF CURSOR RETURN opera%ROWTYPE;
PROCEDURE deschis_ope (ope_var IN OUT ope_tip,
alege IN NUMBER);
END alfa;
CREATE OR REPLACE PACKAGE BODY alfa AS
PROCEDURE deschis_ope (ope_var IN OUT ope_tip,
alege IN NUMBER) IS
BEGIN
IF alege = 1 THEN
OPEN ope_var FOR SELECT * FROM opera;
ELSIF alege = 2 THEN
OPEN ope_var FOR SELECT * FROM opera WHERE valoare>2000;
ELSIF alege = 3 THEN
OPEN ope_var FOR SELECT * FROM opera WHERE valoare=7777;
END IF;
END deschis_ope;
END alfa;
```

Exemplu:

in urmatorul exemplu se declara o variabila cursor care se asociaza unei comenzi *SELECT (SQL* dinamic) ce returneaza anumite linii din tabelul *opera*.

```
DECLARE
TYPE operaref IS REF CURSOR; opera var operaref;
mm_val INTEGER := 100000;
BEGIN
OPEN opera_var FOR
'SELECT cod_opera,valoare FROM opera WHERE valoare> :vv'
USING mm val;
END;
```

Comanda *FETCH* returneaza o linie din mul|imea rezultat a cererii multi-linie, atribuie valori componentelor din lista cererii prin clauza *INTO*, avanseaza cursorul la urmatoarea linie. Sintaxa comenzii este:

FETCH {variabila_cursor | :variabila_cursor_host}
INTO {variabila [*, variabila]... | inregistrare}*
[BULK COLLECT INTO {numecolecfie [*, nume_colecjie]...} |*
{nume array host [*, nume_array_host]...}*
[LIMIT expresie_numerica]];

Clauza *BULK COLLECT* permite incarcarea tuturor liniilor simultan in una sau mai multe colectii. Atributul *nume_colecpe* indica o colec|ie declarata anterior, in care sunt depuse valorile respective, iar *nume_array_host* identifica un vector declarat intr-un mediu gazda *PL/SQL* §i trimis lui *PL/SQL* ca variabila de legatura. Prin clauza *LIMIT* se limiteaza numarul liniilor incarcate din baza de date.

Exemplu:

```
DECLARE
TYPE alfa IS REF CURSOR RETURN opera%ROWTYPE;
TYPE beta IS TABLE OF opera.titlu%TYPE;
TYPE gama IS TABLE OF opera.valoare%TYPE; var1
alfa; var2 beta; var3 gama;
BEGIN
OPEN alfa FOR SELECT titlu, valoare FROM opera;
FETCH var1 BULK COLLECT INTO var2, var3;

CLOSE var1;

END;
```

Comanda *CLOSE* dezactiveaza variabila cursor precizata. Ea are sintaxa:

CLOSE {variabila_cursor | :variabila_cursor_host}

Cursoarele §i variabilele cursor nu sunt interoperabile. Nu poate fi folosita una din ele, cand este a§teptata cealalta. Urmatoarea secven^a este incorecta.

```
DECLARE
TYPE beta IS REF CURSOR RETURN opera%ROWTYPE; gama
beta;
BEGIN
FOR k IN gama LOOP --nu este corect!

END;
```

Expresie cursor

In *Oracle* a fost introdus conceptul de expresie cursor *(cursor expression)*, care returneaza un cursor imbricat *(nested cursor)*.

Expresia cursor are urmatoarea sintaxa:

CURSOR (subcerere)

Fiecare linie din mul|imea rezultat poate confine valori uzuale §i cursoare generate de subcereri. *PL/SQL* accepta cereri care au expresii cursor in cadrul unei declaratii cursor, declaratii *REF CURSOR* §i a variabilelor cursor.

Prin urmare, expresia cursor poate sa apara intr-o comanda *SELECT* ce este utilizata pentru deschiderea unui cursor dinamic. De asemenea, expresiile cursor pot f folosite in cereri *SQL* dinamice sau ca parametri actuali intr-un subprogram.

Un cursor imbricat este incarcat automat atunci cand liniile care il con|in sunt incarcate din cursorul „parinte“ El este inchis daca:

- este inchis explicit de catre utilizator;
- cursorul „parinte“ este reexecutat, inchis sau anulat;
- apare o eroare in timpul unei incarcari din cursorul „parinte“

Exista cateva restric|ii asupra folosirii unei expresii cursor:

- nu poate fi utilizata cu un cursor implicit;
- poate sa apara numai intr-o comanda *SELECT* care nu este imbricata in alta cerere (exceptand cazul in care este o subcerere chiar a expresiei cursor) sau ca argument pentru func|ii tabel, in clauza *FROM* a lui *SELECT*;
- nu poate sa apara in interogarea ce define§te o vizualizare;
- nu se pot efectua opera|ii *BIND* sau *EXECUTE* cu aceste expresii.

Exemplu:

Sa se defineasca un cursor care furnizeaza codurile operelor exp use in cadrul unei expozi|ii avand un cod specificat *(val cod)* §i care se desfa§oara intr-o localitate precizata *(val oras).* Sa se afi§eze data cand a avut loc vernisajul acestei expozi|ii.

in acest caz cursorul returneaza doua coloane, cea de-a doua coloana fiind un cursor imbricat.

```
CURSOR alfa (val_cod NUMBER, val_oras VARCHAR2(20)) IS
  SELECT l.datai,
       CURSOR (SELECT d.cod_expo,
                    CURSOR (SELECT f.cod_opera FROM
                          figureaza_in f WHERE
                          f.cod_expo=d.cod_expo) AS
xx
                FROM expozitie d
                WHERE l.cod_expo = d.cod_expo) AS yy FROM
  locped l
  WHERE    cod_expo = val_cod AND nume_oras= val_oras;
```

Exemplu:

Sa se listeze numele galeriilor din muzeu §i pentru fiecare galerie sa se afi§eze numele salilor din galeria respectiva.

Sunt prezentate doua variante de rezolvare. Prima varianta reprezinta o implementare simpla utilizand programarea secven^iala clasica, iar a doua utilizeaza expresii cursor pentru rezolvarea acestei probleme.

Varianta 1:

```
BEGIN
FOR gal IN (SELECT cod_galerie, nume_galerie FROM
galerie)
LOOP
DBMS_OUTPUT.PUT_LINE (gal.nume_galerie);
FOR sal IN (SELECT cod_sala, nume_sala FROM sala
              WHERE cod_galerie = gal.cod.galerie)
LOOP
DBMS_OUTPUT.PUT_LINE (sal.nume_sala);
END LOOP;
END LOOP;
END;
```

Varianta 2:

```
DECLARE
CURSOR c_gal IS
SELECT nume_galerie,
       CURSOR (SELECT nume_sala FROM sala s
WHERE s.cod_galerie = g.cod_galerie) FROM galerie
g;
v_nume_gal    galerie.nume_galerie%TYPE;
v~sala~    SYS_REFCURSOR;
TYPE sala_nume IS TABLE OF sala.nume_sala%TYPE -
INDEX BY BINARY_INTEGER;
v nume sala sala nume;
BEGIN
OPEN c_gal;
LOOP
FETCH c_gal INTO v_nume_gal, v_sala;
EXIT WHEN c_gal%NOTFOUND;    -
DBMS_OUTPUT.PUT_LINE (v_nume_gal);
FETCH v_sala BULK COLLECT INTO v_nume_sala;
FOR ind IN v_nume_sala.FIRST..v_nume_sala.LAST
LOOP
DBMS_OUTPUT.PUT_LINE (v_nume_sala (ind));
END LOOP;
END LOOP;
CLOSE c_gal;
END;
```