Tema: Gestiunea cursoarelor

Pentru a procesa o comanda *SQL*, sistemul *Oracle* foloseste o zona de memorie cunoscuta sub numele de zona context (*context area*). Când este procesata o instructiune *SQL*, *server*-ul *Oracle* deschide aceasta zona de memorie în care comanda este analizata sintactic si executata.

Zona contine informatii necesare procesarii comenzii, cum ar fi:

- numarul de rânduri procesate de instructiune;
- un *pointer* catre reprezentarea interna a comenzii;
- în cazul unei cereri, multimea rândurilor rezultate în urma executiei acestei comenzi (*active set*).

Un cursor este un *pointer* la aceasta zona context. Prin intermediul cursoarelor, un program *PL/SQL*

poate controla zona context si transformarile petrecute în urma procesarii comenzii.

Exista doua tipuri de cursoare:

- implicite, generate de *server*-ul *Oracle* când în partea executabila a unui bloc *PL/SQL* apare o instructiune *SQL*;
- explicite, declarate si definite de catre utilizator atunci când o cerere (*SELECT*), care apare într-un bloc *PL/SQL*, întoarce mai multe linii ca rezultat.

Atât cursoarele implicite cât si cele explicite au o serie de atribute ale caror valori pot fi folosite în expresii. Lista atributelor este urmatoarea:

- %ROWCOUNT, care este de tip întreg si reprezinta numarul liniilor încarcate de cursor;
- %FOUND, care este de tip boolean si ia valoarea TRUE daca ultima operatie de încarcare (FETCH) dintr-un cursor a avut succes (în cazul cursoarelor explicite) sau daca instructiunea SQL a întors cel putin o linie (în cazul cursoarelor implicite);
- %NOTFOUND, care este de tip boolean si are semnificatie opusa fata de cea a atributului %FOUND;
- %ISOPEN, care este de tip boolean si indica daca un cursor este deschis (în cazul cursoarelor implicite, acest atribut are întotdeauna valoarea FALSE, deoarece un cursor implicit este închis de sistem imediat dupa executarea instructiunii SQL asociate).

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

Cursoare implicite

Când 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 informatii referitoare la ultima comanda *INSERT*, *UPDATE*, *DELETE* sau *SELECT INTO* executata. Înainte ca *Oracle* sa deschida cursorul *SQL* implicit, atributele acestuia au valoarea *null*.

În *Oracle9i*, 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*) contine numarul de linii procesate de a *j*-a executie a unei comenzi *INSERT*, *DELETE* sau *UPDATE*. Daca a *j*-a executie nu afecteaza nici o linie, atunci atributul returneaza valoarea 0. Comanda *FORALL* si atributul

%BULK_ROWCOUNT au aceiasi indici, deci folosesc acelasi domeniu. Daca *%BULK_ROWCOUNT*(*j*) este zero, atributul *%FOUND* este *FALSE*.

***Exemplu*:**

În exemplul care urmeaza, comanda *FORALL* insereaza un numar arbitrar de linii la fiecare iteratie, iar dupa fiecare iteratie 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

Pentru gestiunea cursoarelor explicite sunt necesare urmatoarele etape:

- declararea cursorului (atribuirea unui nume si asocierea cu o comanda SELECT);

- deschiderea cursorului pentru cerere (executarea interogarii asociate si determinarea multimii rezultat);
- recuperarea liniilor rezultatului în variabile PL/SQL;
- închiderea cursorului (eliberarea resurselor relative la cursor).

Prin urmare, pentru a utiliza un cursor, el trebuie declarat în sectiunea declarativa a programului, trebuie deschis în partea executabila, urmând sa fie utilizat apoi pentru extragerea datelor. Daca nu mai este necesar în restul programului, cursorul trebuie sa fie închis.

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

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

Declararea unui cursor explicit

Prin declaratia *CURSOR* în cadrul comenzii *DECLARE* este definit un cursor explicit si 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.

Observatii:

- Comanda SELECT care apare în declararea cursorului, nu trebuie sa includa clauza INTO.
- Daca se cere procesarea liniilor într-o anumita ordine, atunci în cerere este utilizata clauza ORDER BY.
- Variabilele care sunt referite în comanda de selectare trebuie declarate înaintea comenzii CURSOR. Ele sunt considerate variabile de legatura.

- Daca în lista comenzii SELECT apare o expresie, atunci pentru expresia respectiva trebuie utilizat un
- alias, iar câmpul expresie se va referi prin acest alias.
- Numele cursorului este un identificator unic în cadrul blocului, care nu poate sa apara într-o expresie si caruia nu i se poate atribui o valoare.

Deschiderea unui cursor explicit

Comanda *OPEN* executa cererea asociata cursorului, identifica multimea liniilor rezultat si pozitioneaza cursorul înaintea primei linii.

Deschiderea unui cursor se face prin comanda:

OPEN *nume_cursor*;

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 în declaratia cursorului);
- este determinata multimea rezultat (active set) prin executarea cererii SELECT, având în vedere valorile de la pasul anterior;
- pointer-ul este pozitionat la prima linie din multimea activa.

Încarcarea datelor dintr-un cursor explicit

Comanda *FETCH* regaseste liniile rezultatului din multimea activa. *FETCH* realizeaza urmatoarele operatii:

- avanseaza pointer-ul la urmatoarea linie în multimea activa (pointer-ul poate avea doar un sens de deplasare de la prima spre ultima înregistrare);
- citeste datele liniei curente în variabile PL/SQL;
- daca pointer-ul este pozitionat la sfârsitul multimii active atunci se iese din bucla cursorului.

Comanda FETCH are urmatoarea sintaxa:

FETCH *nume_cursor*

INTO {*nume_variabila* [, *nume_variabila*] … | *nume_înregistrare*};

Identificatorul *nume_cursor* reprezinta numele unui cursor declarat si deschis anterior. Variabila sau lista de variabile din clauza *INTO* trebuie sa fie compatibila (ca ordine si tip) cu lista selectata din cererea asociata cursorului.

La un moment dat, comanda *FETCH* regaseste o singura linie. Totusi, în ultimele versiuni *Oracle* pot fi încarcate mai multe linii (la un moment dat) într-o colectie, utilizând clauza *BULK COLLECT*.

***Exemplu*:**

În exemplul care urmeaza se încarca date dintr-un cursor în 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;
```

Închiderea unui cursor explicit

Dupa ce a fost procesata multimea activa, cursorul trebuie închis. Prin aceasta operatie, *PL/SQL* este informat ca programul a terminat folosirea cursorului si resursele asociate acestuia pot fi eliberate. Aceste resurse includ spatiul utilizat pentru memorarea multimii active si spatiul temporar folosit pentru determinarea multimii active.

Cursorul va fi închis prin comanda *CLOSE*, care are urmatoarea sintaxa:

CLOSE *nume_cursor*;

Identificatorul *nume_cursor* este numele unui cursor deschis anterior.

Pentru a reutiliza cursorul este suficient ca acesta sa fie redeschis. Daca se încearca încarcarea datelor dintr-un cursor închis, atunci apare exceptia *INVALID_CURSOR*. Un bloc *PL/SQL* poate sa se termine fara a închide cursoarele, dar acest lucru nu este indicat, deoarece este bine ca resursele sa fie eliberate.

***Exemplu*:**

Pentru toti artistii care au opere de arta expuse în muzeu sa se insereze în tabelul *temp* informatii referitoare la numele acestora si anul nasterii.

```
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 în urmatorul tabel:

		%FOUND	***%ISOPEN***	***%NOTFOUND***	***%ROWCOUNT***
OPEN	Înainte	Exceptie	*False*	Exceptie	Exceptie
	Dupa	*Null*	*True*	*Null*	0
Prima	Înainte	*Null*	*True*	*Null*	0
încarcare	Dupa	*True*	*True*	*False*	1
Urmatoarea	Înainte	*True*	*True*	*False*	1
încarcare	Dupa	*True*	*True*	*False*	Depinde de date
Ultima	Înainte	*True*	*True*	*False*	Depinde de date
încarcare	Dupa	*False*	*True*	*True*	Depinde de date
CLOSE	Înainte	*False*	*True*	*True*	Depinde de date
	Dupa	Exceptie	*False*	Exceptie	Exceptie

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

Într-un pachet poate fi separata specificarea unui cursor de corpul acestuia. Cursorul va fi declarat în specificatia pachetului prin comanda:

CURSOR *nume_cursor* [(*parametru* [, *parametru*]…)]

RETURN *tip_returnat*;

În felul acesta va creste flexibilitatea programului, putând fi modificat doar corpul cursorului, fara a schimba specificatia.

***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 foloseste operatia de ciclare (*LOOP, WHILE, FOR*), prin care la fiecare iteratie se va încarca o noua linie. Comanda *EXIT* poate fi utilizata pentru iesirea din ciclu, iar valoarea atributului *%ROWCOUNT* pentru terminarea ciclului.

Procesarea liniilor unui cursor explicit se poate realiza si cu ajutorul unui ciclu *FOR* special, numit ciclu cursor. Pentru acest ciclu este necesara doar declararea cursorului, operatiile de deschidere, încarcare si închidere ale acestuia fiind implicite.

Comanda are urmatoarea sintaxa:

FOR *nume_înregistrare* ***IN*** *nume_cursor* ***LOOP***

secventa_de_instructiuni;

END LOOP;

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

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

Exemplu:

Sa se calculeze, utilizând un ciclu cursor cu subcereri, valoarea operelor de arta expuse într-o galerie al carei cod este introdus de la tastatura. De asemenea, sa se obtina media valorilor operelor de arta expuse în 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 NUMBER; i INTEGER;
BEGIN
val:=0; i:=0;
FOR numar_opera IN
(SELECT         cod_opera, valoare FROMopera
```

```
     WHERE           cod_galerie = v_cod_galerie) LOOP val := val +
numar_opera.valoare;
     i := i+1;
     END LOOP;--închidere 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 îi corespunde o comanda *SELECT*, care nu poate fi schimbata pe parcursul programului. Pentru a putea lucra cu niste cursoare ale caror comenzi *SELECT* atasate depind de parametri ce pot fi modificati la momentul executiei, în *PL/SQL* s-a introdus notiunea de cursor parametrizat. Prin urmare, un cursor parametrizat este un cursor în care comanda *SELECT* atasata depinde de unul sau mai multi parametri.

Transmiterea de parametri unui cursor parametrizat se face în mod similar procedurilor stocate. Un astfel de cursor este mult mai usor de interpretat si de întretinut, oferind si posibilitatea reutilizarii sale în blocul *PL/SQL*.

Declararea unui astfel de cursor se face respectând urmatoarea sintaxa:

CURSOR *nume_cursor* [(*nume_parametru*[, *nume_parametru ...*])]

[***RETURN*** *tip_returnat*]

IS *comanda_select*;

Identificatorul *comanda_select* este o instructiune *SELECT* fara clauza *INTO*, *tip_returnat* reprezinta un tip înregistrare sau linie de tabel, iar *nume_parametru* are sintaxa:

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

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

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

Sintaxa pentru deschiderea unui cursor parametrizat este:

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

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

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

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

***Exemplu*:**

Utilizând un cursor parametrizat sa se obtina codurile operelor de arta din fiecare sala, identificatorul salii si al galeriei. Rezultatele sa fie inserate în 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 înainte ca acestea sa fie sterse sau reactualizate. Blocarea se poate realiza (atunci când cursorul este deschis) cu ajutorul comenzii *SELECT* care contine clauza *FOR UPDATE*.

Declararea unui astfel de cursor se face conform sintaxei:

CURSOR *nume_cursor* ***IS***

comanda_select

FOR UPDATE [***OF*** *lista_câmpuri*] [***NOWAIT***];

Identificatorul *lista_câmpuri* este o lista ce include câmpurile 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.

În momentul deschiderii unui astfel de cursor, liniile corespunzatoare multimii active, determinate de clauza *SELECT*, sunt blocate pentru operatii de scriere (reactualizare sau stergere). În felul acesta este realizata consistenta la citire a sistemului. De exemplu, aceasta situatie este utila când se reactualizeaza o valoare a unei linii si trebuie avuta siguranta ca linia nu este schimbata de alt utilizator înaintea reactualizarii. Prin urmare, alte sesiuni nu pot schimba liniile din multimea activa pâna când tranzactia nu este permanentizata sau anulata. Daca alta sesiune a blocat deja liniile din multimea activa, atunci comanda *SELECT ... FOR UPDATE* va astepta (sau nu) ca aceste blocari sa fie eliberate. Pentru a trata aceasta situatie se utilizeaza clauza *WAIT*, respectiv *NOWAIT*.

În *Oracle9i* este utilizata sintaxa:

SELECT *...* ***FROM*** *...* ***FOR UPDATE*** [***OF*** *lista_campuri*] [{***WAIT*** *n* | ***NOWAIT***}];

Valoarea lui *n* reprezinta numarul de secunde de asteptare. Daca liniile nu sunt deblocate în *n* secunde, atunci se declanseaza 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 asteapta pâna ce linia este deblocata si atunci returneaza rezultatul comenzii

SELECT.

Daca un cursor este declarat cu clauza *FOR UPDATE*, atunci comenzile *DELETE* si *UPDATE* corespunzatoare trebuie sa contina clauza *WHERE CURRENT OF nume_cursor*.

Aceasta clauza refera linia curenta care a fost gasita de cursor, permitând ca reactualizarile si stergerile sa se efectueze asupra acestei linii, fara referirea explicita a cheii primare sau pseudocoloanei *ROWID*. De subliniat ca instructiunile *UPDATE* si *DELETE* vor reactualiza numai coloanele listate în clauza *FOR UPDATE*.

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

Dupa închiderea cursorului este necesara comanda *COMMIT* pentru a realiza scrierea efectiva a modificarilor, deoarece cursorul lucreaza doar cu niste copii ale liniilor reale existente în tabele.

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

***Exemplu*:**

Sa se dubleze valoarea operelor de arta pictate pe pânza care au fost achizitionate înainte 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 în momentul în care blocul este compilat.

În *PL/SQL* a fost introdusa variabila cursor, care este de tip referinta. 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.

În momentul declararii, variabilele cursor nu solicita o comanda *SQL* asociata. În acest fel, diferite comenzi *SQL* pot fi asociate variabilelor cursor, la diferite momente de timp. Acest tip de variabila trebuie declarata, deschisa, încarcata si închisa în mod similar unui cursor static.

Variabilele cursor sunt dinamice deoarece li se pot asocia diferite interogari atâta timp cât coloanele returnate de fiecare interogare corespund declaratiei variabilei cursor.

Aceste variabile sunt utile în transmiterea seturilor de rezultate între subprograme *PL/SQL* stocate si diferiti clienti. De exemplu, un *client OCI*, o aplicatie *Oracle Forms* si *server*-ul *Oracle* pot referi aceeasi zona de lucru (care contine multimea rezultat). Pentru a reduce traficul în retea, o variabila cursor poate fi declarata pe statia *client*, deschisa si se pot încarca date din ea pe *server*, apoi poate continua încarcarea, dar de pe statia *client* etc.

Pentru a crea o variabila cursor este necesara definirea unui tip *REF CURSOR*, urmând 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 în declaratiile urmatoare ale variabilelor cursor, iar *tip_returnat* este un tip înregistrare 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 lipseste clauza *RETURN*, cursorul poate fi deschis pentru orice cerere *SELECT*.

Daca variabila cursor apare ca parametru într-un subprogram, atunci trebuie specificat tipul parametrului (tipul *REF CURSOR*) si forma acestuia (*IN* sau *IN OUT*).

Exista anumite restrictii referitoare la utilizarea variabilelor cursor:

- nu pot fi declarate într-un pachet;
- cererea asociata variabilei cursor nu poate include clauza FOR UPDATE (restrictia dispare în *Oracle9i*);
- nu poate fi asignata valoarea null unei variabile cursor;
- nu poate fi utilizat tipul REF CURSOR pentru a specifica tipul unei coloane în comanda CREATE TABLE;
- nu pot fi utilizati operatorii de comparare pentru a testa egalitatea, inegalitatea sau valoarea null a variabilelor cursor;

- nu poate fi utilizat tipul REF CURSOR pentru a specifica tipul elementelor unei colectii (varray, nested table);
- nu pot fi folosite cu SQL dinamic în Pro*C/C++.

În cazul variabilelor cursor, instructiunile de deschidere (*OPEN*), încarcare (*FETCH*), închidere (*CLOSE*) vor avea o sintaxa similara celor comentate anterior.

Comanda *OPEN...FOR* asociaza o variabila cursor cu o cerere multilinie, executa cererea, identifica multimea rezultat si pozitioneaza cursorul la prima linie din multimea rezultat. Sintaxa comenzii este:

OPEN {*variabila_cursor* | :*variabila_cursor_host*}

FOR {*cerere_select* |

sir_dinamic [***USING*** *argument_bind* [*, argument_bind ...*]] };

Identificatorul *variabila_cursor* specifica o variabila cursor declarata anterior, dar fara optiunea *RETURN tip*, *cerere_select* este interogarea pentru care este deschisa variabila cursor, iar *sir_dinamic* este o secventa de caractere care reprezinta cererea multilinie.

Optiunea *sir_dinamic* este specifica prelucrarii dinamice a comenzilor, iar posibilitatile oferite de *SQL* dinamic vor fi analizate într-un capitol separat. Identificatorul :*variabila_cursor_host* reprezinta o variabila cursor declarata într-un mediu gazda *PL/SQL* (de exemplu, un program *OCI*).

Comanda *OPEN...FOR* poate deschide acelasi cursor pentru diferite cereri. Nu este necesara închiderea variabilei cursor înainte 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 > 200;
ELSIF alege = 3 THEN
OPEN ope_var FOR SELECT * FROM opera WHERE valoare = 777;
```

```
END IF;
END deschis_ope;
END alfa;
```

Comanda *FETCH* returneaza o linie din multimea rezultat a cererii, atribuie valorile returnate de cerere componentelor din lista specificata prin clauza *INTO* si avanseaza cursorul la urmatoarea linie. Comanda are urmatoarea sintaxa:

FETCH {*variabila_cursor* | :*variabila_cursor_host*}

INTO {*variabila* [, *variabila ...*] | *înregistrare*}

[***BULK COLLECT INTO*** {*nume_colectie* [, *nume_colectie ...*] } |

{*nume_array_host* [, *nume_array_host ...*] } [***LIMIT*** *expresie_numerica*]];

Clauza *BULK COLLECT INTO* permite încarcarea tuturor liniilor simultan în una sau mai multe colectii. Atributul *nume_colectie* indica o colectie declarata anterior, în care sunt depuse valorile respective, iar *nume_array_host* identifica un vector declarat într-un mediu gazda *PL/SQL* si trimis lui *PL/SQL* ca variabila de legatura. Prin clauza *LIMIT* se limiteaza numarul liniilor încarcate 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 var1 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 si variabilele cursor nu sunt interoperabile. Nu poate fi folosita una dintre ele, atunci când este asteptata cealalta.

Expresie cursor

În *Oracle9i* 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 multimea rezultat poate contine valori uzuale si cursoare generate de subcereri. *PL/SQL* accepta cereri care au expresii cursor în cadrul unei declaratii cursor, declaratii *REF CURSOR* si a variabilelor cursor.

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

Un cursor imbricat este încarcat automat atunci când liniile care îl contin sunt încarcate din cursorul „parinte“. El este închis daca:

- este închis explicit de catre utilizator;
- cursorul „parinte“ este reexecutat, închis sau anulat;
- apare o eroare în timpul unei încarcari din cursorul „parinte“.
- Exista câteva restrictii asupra folosirii unei expresii cursor:
- nu poate fi utilizata cu un cursor implicit;
- poate sa apara numai într-o comanda SELECT care nu este imbricata în alta cerere (exceptând cazul în care este o subcerere chiar a expresiei cursor) sau ca argument pentru functii tabel, în clauza FROM a lui SELECT;
- nu poate sa apara în interogarea ce defineste o vizualizare;
- nu se pot efectua operatii BIND sau EXECUTE cu aceste expresii.

***Exemplu*:**

Sa se defineasca un cursor care furnizeaza codurile operelor expuse în cadrul unei expozitii având un cod specificat (*val_cod*) si care se desfasoara într-o localitate precizata (*val_oras*). Sa se afiseze data când a avut loc vernisajul acestei expozitii.

În 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 FROMexpozitie 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 si pentru fiecare galerie sa se afiseze numele salilor din galeria respectiva.

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

Varianta 1:

```
BEGIN
FOR gal IN (SELECT cod_galerie, nume_galerie FROMgalerie)
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_salaSYS.REFCURSOR;
TYPE sala_nume IS TABLE OF sala.nume_sala%TYPE
INDEX BY BINARY_INTEGER;
v_nume_salasala_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;
```