1. Tipuri de date în *PL/SQL*

Fiecare variabila sau constanta utilizata într-un bloc *PL/SQL* este de un anumit tip care determina formatul sau de stocare, constrângerile pe care trebuie sa le verifice si domeniul valorilor sale.

Variabilele folosite în Oracle9i pot fi împartite în doua clase:

- variabile specifice PL/SQL, care se clasifica în variabile de tip scalar, compuse, referinta, LOB (large objects) si tipuri obiect;
- variabile nespecifice PL/SQL, care pot fi variabile de legatura (bind variables), variabile gazda (host variables) si variabile indicator.

Variabile specifice *PL/SQL*

Tipurile de date scalare nu au componente interne (contin valori atomice). Aceste tipuri de date se împart în cinci clase fundamentale.

- Tipurile de date ce stocheaza valori numerice, cuprind: tipul *NUMBER* cu subtipurile *DEC*, *DECIMAL*, *DOUBLE PRECISION*, *INT*, *INTEGER*, *NUMERIC*, *FLOAT*, *REAL*, *SMALLINT*; tipul de date *BINARY_INTEGER* cu subtipurile *NATURAL*, *NATURALN*, *POSITIVE*, *POSITIVEN*, *SIGNTYPE*; tipul *PLS_INTEGER*..
- Tipurile de date ce stocheaza caractere, cuprind: tipul *VARCHAR2* cu subtipurile *STRING*, *VARCHAR*; tipul de date *CHAR* cu subtipul *CHARACTER*; tipurile *LONG*, *RAW*, *LONG RAW*, *ROWID*.
- Tipurile de date ce stocheaza data calendaristica si ora, cuprind tipurile *DATE*, *TIMESTAMP*, *TIMESTAMP WITH TIME ZONE*, *TIMESTAMP WITH LOCAL TIME ZONE*, *INTERVAL YEAR TO MONTH*, *INTERVAL DAY TO SECOND*.
- Tipurile de date pentru globalizare, care includ tipurile *NCHAR* si *NVARCHAR2*, stocheaza date în format *Unicode*.
- Tipul de date *BOOLEAN* stocheaza valori logice (*true*, *false* sau *null*).

Tipurile de date compuse au componente interne care pot fi prelucrate individual. Sistemul ofera programatorului doua tipuri de date compuse: înregistrare (*RECORD*) si colectie (*INDEX-BY TABLE*, *NESTED TABLE*, *VARRAY*).

Tipurile de date referinta (*REF CURSOR*, *REF obiect*) sunt tipuri de date ale caror valori, numite *pointer*-i, fac referinta catre obiecte din program. *Pointer*-ii contin locatia de memorie (adresa) a unui element si nu elementul în sine. Tipul *REF CURSOR* este folosit pentru a face referinta la un cursor explicit. Tipul *REF obiect* face referinta la adresa unui obiect.

Tipurile *LOB* (*large object*) sunt acele tipuri de date ale caror valori, numite locatori (*locators*),

specifica locatia (localizarea) unor obiecte de dimensiuni mari, adica blocuri de date nestructurate, cum ar fi texte, imagini grafice, clipuri video si sunete. Ele cuprind tipurile: *CLOB* (*Character Large Object*), *BLOB* (*Binary Large Object*), *NCLOB* (*National Language Character Large Object*) si *BFILE* (*Binary File*). Tipurile *LOB* sunt prelucrate cu ajutorul pachetului *DBMS_LOB*.

Tipurile obiect sunt tipuri compuse, definite de utilizator, care încapsuleaza structuri de date (atribute) împreuna cu subprograme pentru prelucrarea datelor (metode). *Oracle9i* extinde modelul obiect de la versiunea *Oracle8i*, implementând mostenirea prin intermediul subtipurilor. Tipul obiect va fi analizat într-un capitol separat.

Dintre tipurile scalare *PL/SQL*, urmatoarele sunt si tipuri *SQL* (adica pot fi folosite pentru coloanele tabelelor *Oracle*): *NUMBER, VARCHAR2, CHAR, LONG, RAW, LONG RAW, ROWID, NCHAR, NVARCHAR2, DATE*. În unele cazuri, tipurile de date *PL/SQL* difera de corespondentele lor din *SQL* prin dimensiunea maxima permisa.

Tipul *NUMBER* memoreaza numerele în virgula fixa si virgula mobila. El are forma generala *NUMBER* (*m, n*), unde *m* reprezinta numarul total de cifre, iar *n* numarul de zecimale. Valoarea unei variabile de tip *NUMBER* este cuprinsa între 1.0E-129 si 9.99E125. Numarul de zecimale determina pozitia în care apare rotunjirea. Valoarea sa este cuprinsa între -84 si 127, iar implicit este 0.

Tipul *NUMBER* are urmatoarele subtipuri, care au aceleasi intervale de valori: *NUMERIC, REAL, DEC, DECIMAL* si *DOUBLE PRECISION* (pentru memorarea datelor numerice în virgula fixa), *FLOAT* (pentru memorarea datelor numerice în virgula mobila), *SMALLINT, INTEGER* si *INT* (pentru memorarea numerelor întregi). Aceste subtipuri se pot utiliza pentru compatibilitate *ANSI/ISO, IBM SQL/DS* sau *IBM DB2*.

Tipul *BINARY_INTEGER* memoreaza numere întregi cu semn având valori cuprinse între $-2^{31} - 1$ si 2^{31} - Acest tip de date este utilizat frecvent pentru indecsii tabelelor, nu necesita conversii si admite mai multe subtipuri. De exemplu, pentru a restrictiona domeniul variabilelor la valori întregi nenegative se utilizeaza tipurile *NATURAL* ($0 .. 2^{31} – 1$) si *POSITIVE* ($1 .. 2^{31} – 1$).

Tipul *PLS_INTEGER* este utilizat pentru stocarea numerelor întregi cu semn si are acelasi interval de definire ca si tipul *BINARY_INTEGER*. Operatiile cu acest tip sunt efectuate mai rapid (folosesc aritmetica masinii), decât cele cu tipurile *NUMBER* sau *BINARY_INTEGER* (folosesc librarii aritmetice). Prin urmare, pentru o mai buna performanta, este preferabil sa se utilizeze tipul *PLS_INTEGER*.

Variabilele alfanumerice pot fi de tip *CHAR, VARCHAR2, LONG, RAW* si *LONGRAW*. Reprezentarea interna depinde de setul de caractere ales (*ASCII* sau *EBCDIC*).

Tipurile *CHAR, VARCHAR2* si *RAW* pot avea un parametru pentru a preciza lungimea maxima. Daca aceasta nu este precizata atunci, implicit, se considera 1. Lungimea este exprimata în octeti (nu în caractere). Subtipurile acestor tipuri se pot utiliza pentru compatibilitate *ANSI/ISO, IBM SQL/DS* sau *IBM DB2*.

În *Oracle9i* a fost extinsa sintaxa pentru *CHAR* si *VARCHAR2*, permitând ca variabila ce precizeaza lungimea maxima sa fie de tip *CHAR* sau *BYTE*.

Variabilele de tip *LONG* pot memora texte, tabele de caractere sau documente, prin urmare siruri de caractere de lungime variabila de pâna la 32760 octeti. Este similar tipului *VARCHAR2*.

Tipul *RAW* permite memorarea datelor binare (biti) sau a sirurilor de octeti. De exemplu, o variabila *RAW* poate memora o secventa de caractere grafice sau o imagine digitizata. Tipul *RAW* este similar tipului alfanumeric, cu exceptia faptului ca *PL/SQL* nu interpreteaza datele de tip *RAW*. *Oracle* nu face conversia datelor de acest tip, atunci când se transmit de la un sistem la altul. Chiar daca lungimea maxima a unei variabile *RAW* poate fi 32767 octeti, într-o coloana *RAW* a bazei de date nu se pot introduce decât 2000 octeti. Pentru a insera valori mai mari se foloseste o coloana de tip *LONG RAW*, care are lungimea maxima 2^{31} octeti. *LONG RAW* este similar tipului *LONG*, dar datele nu mai sunt interpretate de *PL/SQL*.

Tipurile *TIMESTAMP*, *TIMESTAMP WITH TIME ZONE*, *TIMESTAMP WITH LOCAL TIME ZONE*,

INTERVAL YEAR TO MONTH, INTERVAL DAY TO SECOND au fost introduse în *Oracle9i* si permit rafinari ale tipului *DATE*. De exemplu, *TIMESTAMP* poate lua în considerare si fractiuni de secunda.

PL/SQL suporta doua seturi de caractere: una specifica bazei de date care este utilizata pentru definirea identificatorilor si a codului sursa (*database character set - DCS*) si o multime de caractere nationale care este folosita pentru reprezentarea informatiei cu caracter national (*national character set - NCS*).

Tipurile de date *NCHAR* si *NVARCHAR2* sunt utilizate pentru stocarea în baza de date a sirurilor de caractere ce folosesc *NCS*. Ele ofera suport pentru globalizarea datelor, astfel încât utilizatorii din toata lumea pot interactiona cu *Oracle* în limba lor nationala. Aceste tipuri de date suporta numai date *Unicode*.

Unicode este o multime de caractere globale care permite stocarea de informatie în orice limba, folosind o multime unica de caractere. Prin urmare, *unicode* furnizeaza o valoare cod unica pentru fiecare caracter, indiferent de platforma, program sau limba.

Variabile nespecifice *PL/SQL*

Variabila de legatura (*bind*) se declara într-un mediu gazda si este folosita pentru transferul (la executie) valorilor numerice sau de tip caracter în/din unul sau mai multe programe *PL/SQL*. Variabilele declarate în mediul gazda sau în cel apelant pot fi referite în instructiuni *PL/SQL*, daca acestea nu sunt în cadrul unei proceduri, functii sau pachet.

În *SQL*Plus*, variabilele de legatura se declara folosind comanda *VARIABLE*, iar pentru afisarea

valorilor acestora se utilizeaza comanda *PRINT*. Ele sunt referite prin prefixarea cu simbolul „:“, pentru a putea fi deosebite de variabilele declarate în *PL/SQL*.

Deoarece instructiunile *SQL* pot fi integrate în programe *C*, este necesar un mecanism pentru a transfera valori între mediul de programare *C* si instructiunile *SQL* care comunica cu *server*-ul bazei de date *Oracle*. În acest scop, în programul încapsulat sunt definite variabilele gazda (*host*). Acestea sunt declarate între directivele *BEGIN DECLARE SECTION* si *END DECLARE SECTION* ale preprocesorului.

O valoare *null* în baza de date nu are o valoare corespunzatoare în mediul limbajului gazda (de exemplu, limbajul *C*). Pentru a rezolva problema comunicarii valorilor *null* între programul scris în limbaj gazda si sistemul *Oracle*, au fost definite variabilele indicator. Acestea sunt variabile speciale de tip întreg, folosite pentru a indica daca o valoare *null* este recuperata (extrasa) din baza de date sau stocata în aceasta. Ele au urmatoarea forma:

:*nume_extern* [: *indicator*]

De exemplu, daca atribuirea este facuta de limbajul gazda, valoarea –1 a indicatorului specifica faptul ca *PL/SQL* trebuie sa înlocuiasca valoarea variabilei prin *null*, iar o valoare a indicatorului mai mare ca zero precizeaza ca *PL/SQL* trebuie sa considere chiar valoarea variabilei.

Declararea variabilelor

Identificatorii *PL/SQL* trebuie declarati înainte de a fi referiti în blocul *PL/SQL*. Daca în declaratia unei variabile apar referiri la alte variabile, acestea trebuie sa fi fost declarate anterior. Orice variabila declarata într-un bloc este accesibila blocurilor continute sintactic în acesta.

Tipurile scalare sunt predefinite în pachetul *STANDARD*. Pentru a folosi un astfel de tip într-un program este suficient sa fie declarata o variabila de tipul respectiv.

Tipurile compuse sunt definite de utilizator. Prin urmare, în acest caz trebuie definit efectiv tipul si apoi declarata variabila de tipul respectiv.

În declararea variabilelor pot fi utilizate atributele *%TYPE* si *%ROWTYPE*, care reprezinta tipuri de date implicite. Aceste tipuri permit declararea unei variabile în concordanta cu declaratii de variabile facute anterior.

Atributul *%TYPE* permite definirea unei variabile având tipul unei variabile declarate anterior sau tipul unei coloane dintr-un tabel.

Atributul *%ROWTYPE* permite definirea unei variabile având tipul unei înregistrari dintr-un tabel. Avantajul utilizarii acestui atribut consta în faptul ca nu este necesar sa se cunoasca numarul si tipurile coloanelor tabelului. Elementele individuale ale acestei structuri de tip înregistrare sunt referite în maniera clasica, prefixând numele coloanei cu numele variabilei declarate.

Calitatea atributelor *%TYPE* si *%ROWTYPE* consta în faptul ca simplifica întretinerea codului

PL/SQL. De exemplu, poate fi modificata dimensiunea unei coloane, fara sa fie necesara modificarea declaratiei variabilelor al caror tip s-a definit facând referinta la tipul coloanei respective.

Sintaxa declararii unei variabile este urmatoarea:

identificator [***CONSTANT***] {*tip_de_date* | *identificator****%TYPE*** |

*identificator****%ROWTYPE***} [***NOT NULL***] [{:= | ***DEFAULT***} *expresie_PL/SQL*];

Se pot defini constante (valoarea stocata nu poate fi modificata) prin specificarea la declarare a cuvântului cheie *CONSTANT*.

***Exemplu*:**

```
v_valoare          NUMBER(15) NOT NULL := 0;
v_data_achizitie DATE DEFAULT SYSDATE;
v_material         VARCHAR2(15) := 'Matase';
c_valoare          CONSTANT NUMBER := 100000;
v_stare            VARCHAR2(20) DEFAULT 'Buna';
v_clasificare      BOOLEAN  DEFAULT FALSE;
v_cod_opera        opera.cod_opera%TYPE;
v_opera            opera%ROWTYPE;
int_an_luna        INTERVAL YEAR TO MONTH := INTERVAL '3-2' YEAR
TO MONTH;
```

***Observatii*:**

- Pentru a denumi o variabila este utilizata frecvent (pentru usurinta referirii) prefixarea cu litera *v* (*v_identificator*), iar pentru o constanta este folosita prefixarea cu litera *c* (*c_identificator*).
- Variabilele pot fi initializate, iar daca o variabila nu este initializata, valoarea implicita a acesteia este *null*. Daca o variabila este declarata *NOT NULL*, atunci ea va fi obligatoriu initializata.
- Pentru a initializa o variabila sau o constanta poate fi utilizata o expresie *PL/SQL* compatibila ca tip cu variabila sau constanta respectiva.
- Constantele trebuie initializate când sunt declarate, altfel apare o eroare la compilare.
- În sectiunea declarativa, pe fiecare linie, exista o singura declaratie de variabila.
- Doua obiecte (variabile) pot avea acelasi nume cu conditia sa fie definite în blocuri diferite. Daca ele coexista, poate fi folosit doar obiectul declarat în blocul curent.
- Atributul *%ROWTYPE* nu poate include clauze de initializare.

Definirea subtipurilor

Subtipurile deriva dintr-un tip de baza, la care se adauga anumite restrictii. De exemplu, *NATURAL* este un subtip predefinit *PL/SQL*, derivat din tipul de baza *BINARY_INTEGER*, cu restrictia ca permite prelucrarea valorilor întregi nenegative.

Prin urmare, un subtip nu reprezinta un nou tip de date, ci un tip existent asupra caruia se aplica anumite constrângeri. Subtipurile presupun acelasi set de operatii ca si tipul de baza, dar aplicate unui subset de valori al acestui tip.

Sistemul *Oracle* permite ca utilizatorul sa-si defineasca propriile sale tipuri si subtipuri de date în partea declarativa a unui bloc *PL/SQL*, subprogram sau pachet utilizând sintaxa:

SUBTYPE *nume_subtip* ***IS*** *tip_de_baza* [***NOT NULL***];

În dictionarul datelor exista vizualizari care furnizeaza informatii despre tipurile de date create de utilizator (*USER_TYPES, USER_TYPE_ATTRS*).

Conversii între tipuri de date

Exista doua tipuri de conversii, implicite si explicite. *PL/SQL* face automat conversii implicite între caractere si numere sau între caractere si date calendaristice. Chiar daca sistemul realizeaza automat aceste conversii, în practica se utilizeaza frecvent functii de conversie explicita.

Functiile de conversie explicita din *SQL* sunt utilizabile si în *PL/SQL*. *A*cestea sunt: *TO_NUMBER, TO_CHAR, TO_DATE, TO_MULTI_BYTE, TO_SINGLE_BYTE, TO_CLOB, TO_LOB, CHARTOROWID, ROWIDTOCHAR, RAWTOHEX, HEXTORAW*.

În *Oracle9i* se pot folosi urmatoarele functii de conversie: *ASCIISTR, BIN_TO_NUM, NUMTODSINTERVAL, TO_TIMESTAMP, TO_YMINTERVAL, TO_NCHAR, TO_NCLOB, TO_TIMESTAMP_TZ, NUMTOYMINTERVAL, TO_DSINTERVAL, REFTOHEX, RAWTOHEX, RAWTONHEX, FROM_TZ, ROWIDTONCHAR, COMPOSE, DECOMPOSE*.

Denumirile acestor functii reflecta posibilitatile pe care le ofera. De exemplu, *TO_YMINTERVAL* converteste argumentele sale la tipul *INTERVAL YEAR TO MONTH* conform unui format specificat. Functia *COMPOSE* converteste un sir de caractere la un sir *unicode* (asociaza o valoare cod unica pentru fiecare simbol din sir).

Înregistrari

Tipul *RECORD* ofera un mecanism pentru prelucrarea înregistrarilor. Înregistrarile au mai multe câmpuri ce pot fi de tipuri diferite, dar care sunt legate din punct de vedere logic.

Declararea tipului *RECORD* se face conform urmatoarei sintaxe:

TYPE *nume_tip* ***IS RECORD***

(*nume_câmp1* {*tip_câmp* | *variabila****%TYPE*** | *nume_tabel.coloana****%TYPE*** | *nume_tabel****%ROWTYPE***} [[***NOT NULL***] {:= | ***DEFAULT***} *expresie1*],

(*nume_câmp2* {*tip_câmp* | *variabila****%TYPE*** | *nume_tabel.coloana****%TYPE*** | *nume_tabel****%ROWTYPE***} [[***NOT NULL***] {:= | ***DEFAULT***} *expresie2*],…);

Identificatorul *nume_tip* reprezinta numele tipului *RECORD* care se va specifica în declararea înregistrarilor, *nume_câmp* este numele unui câmp al înregistrarii, iar *tip_câmp* este tipul de date al

câmpului.

***Observatii*:**

- Daca un câmp nu este initializat, atunci se considera implicit ca are valoarea *null*. Daca s-a specificat constrângerea *NOT NULL*, atunci obligatoriu câmpul trebuie initializat, iar initializarea se face cu o valoare diferita de *null*.
- Pentru referirea câmpurilor individuale din înregistrare se prefixeaza numele câmpului cu numele înregistrarii.
- Pot fi asignate valori unei înregistrari utilizând comenzile *SELECT*, *FETCH* sau instructiunea clasica de atribuire. De asemenea, o înregistrare poate fi asignata altei înregistrari de acelasi tip.
- Componentele unei înregistrari pot fi de tip scalar, *RECORD*, *TABLE*, obiect, colectie (dar nu de tipul *REF CURSOR*).
- *PL/SQL* permite declararea si referirea înregistrarilor imbricate.
- Numarul de câmpuri ale unei înregistrari nu este limitat.
- Înregistrarile nu pot fi comparate (egalitate, inegalitate sau *null*).
- Înregistrarile pot fi parametri în subprograme si pot sa apara în clauza *RETURN* a unei functii.

Diferenta dintre atributul *%ROWTYPE* si tipul de date compus *RECORD* este ca tipul *RECORD* permite specificarea tipului de date pentru câmpuri si declararea acestora. Numele câmpului poate coincide cu numele unei coloane.

Oracle9i introduce câteva facilitati legate de acest tip de date.

- Se poate insera (*INSERT*) o linie într-un tabel utilizând o înregistrare. Nu mai este necesara listarea câmpurilor individuale, ci este suficienta utilizarea numelui înregistrarii.
- Se poate reactualiza (*UPDATE*) o linie a unui tabel utilizând o înregistrare. Sintaxa *SET ROW*
- permite sa se reactualizeze întreaga linie folosind continutul unei înregistrari.
- Într-o înregistrare se poate regasi si returna sau sterge informatia din clauza *RETURNING* a comenzilor *UPDATE* sau *DELETE*.
- Daca în comenzile *UPDATE* sau *DELETE* se modifica mai multe linii, atunci pot fi utilizate în sintaxa *BULK COLLECT INTO*, colectii de înregistrari.

***Exemplu*:**

Exemplul urmator arata modul în care poate sa fie utilizata o înregistrare în clauza *RETURNING* asociata comenzii *DELETE*.

```
DECLARE
TYPE val_opera IS RECORD ( cheie   NUMBER,
val     NUMBER);
```

```
v_info_valoareval_opera;
BEGIN
DELETE FROM opera
WHERE cod_opera = 753 RETURNING cod_opera, valoare
INTO v_info_valoare;
… END;
```

În *PL/SQL* este folosit frecvent tipul tablou de înregistrari. Referirea la un element al tabloului se face prin forma clasica: *tabel*(*index*).*câmp*.

***Exemplu*:**

Sa se defineasca un tablou de înregistrari având tipul celor din tabelul *organizator*. Sa se initializeze un element al tabloului si sa se introduca în tabelul *organizator*. Sa se stearga elementele tabloului.

```
DECLARE
TYPE org_table_type IS TABLE OF organizator%ROWTYPE INDEX BY
BINARY INTEGER;
org_tableorg_table_type; i   NUMBER;
BEGIN
IF org_table.COUNT <>0 THEN i := org_table.LAST+1; ELSE i:=1;
END IF;
org_table(i).cod_org := 752; org_table(i).nume := 'Grigore
Ion';
org_table(i).adresa := 'Calea Plevnei 18 Sibiu';
org_table(i).tip := 'persoana fizica';
INSERT INTO organizator
VALUES (org_table(i).cod_org, org_table(i).nume,
org_table(i).adresa, org_table(i).tip);
-- sau folosind noua facilitate Oracle9i
-- INSERT INTO organizator
-- VALUES (org_table(i)); org_table.DELETE;
DBMS_OUTPUT.PUT_LINE('Dupa aplicarea metodei DELETE sunt '
||TO_CHAR(org_table.COUNT)||' elemente');
END;
```

Colectii

Uneori este preferabil sa fie prelucrate simultan mai multe variabile de acelasi tip. Tipurile de date care permit acest lucru sunt colectiile. Fiecare element are un indice unic, care determina pozitia sa

în colectie.

Oracle7 a furnizat tipul *index-by table*, initial numit *PL/SQL table* datorita asemanarii sale cu structura tabelelor relationale. *Oracle8* a introdus doua tipuri colectie, *nested table* si *varray*. *Oracle9i* permite crearea de colectii pe mai multe niveluri, adica colectii de colectii.

Prin urmare, în *PL/SQL* exista trei tipuri de colectii:

- tablouri indexate (*index-by tables*);
- tablouri imbricate (*nested tables*);
- vectori (*varrays* sau *varying arrays*).

Tipul *index-by table* poate fi utilizat numai în declaratii *PL/SQL*. Tipurile *varray* si *nested table* pot fi utilizate atât în declaratii *PL/SQL*, cât si în declaratii la nivelul schemei (de exemplu, pentru definirea tipului unei coloane a unui tabel relational).

***Exemplu*:**

În exemplul care urmeaza sunt ilustrate cele trei tipuri de colectii.

```
DECLARE
TYPE tab_index IS TABLE OF NUMBER INDEX BY BINARY_INTEGER;
TYPE tab_imbri IS TABLE OF NUMBER; TYPE vector IS VARRAY(15) OF
NUMBER;
v_tab_index tab_index; v_tab_imbri tab_imbri; v_vector  vector;
BEGIN
v_tab_index(1) := 72;
v_tab_index(2) := 23;
v_tab_imbri := tab_imbri(5, 3, 2, 8, 7); v_vector := vector(1,
2);
END;
```

***Observatii*:**

- Deoarece colectiile nu pot fi comparate (egalitate sau inegalitate), ele nu pot sa apara în clauzele

DISTINCT, GROUP BY, ORDER BY.

- Tipul colectie poate fi definit într-un pachet.
- Tipul colectie poate sa apara în clauza *RETURN* a unei functii.
- Colectiile pot fi parametri formali într-un subprogram.
- Accesul la elementele individuale ale unei colectii se face prin utilizarea unui indice.

Tablouri indexate

Tipul de date *index-by table* ofera un mecanism pentru prelucrarea tablourilor. Tabloul indexat *PL/SQL* are doua componente: o coloana ce cuprinde cheia primara pentru acces la liniile tabloului si o

coloana care include valoarea efectiva a elementelor tabloului.

Oracle7 asigura definirea tablourilor de înregistrari care pot fi declarate si utilizate numai în programe *PL/SQL*, *Oracle8* realizeaza definirea tablourilor de tipuri obiect, iar *Oracle9i* permite definirea tablourilor de colectii.

În *Oracle9i* tipul *index-by table* este redenumit *associative array* pentru compatibilitate (de limbaj) cu termenul folosit în alte limbaje de programare (*C++*, *JavaScript*, *Perl*) pentru a defini aceasta structura de date.

Tablourile indexate *PL/SQL* trebuie definite în doi pasi: se defineste tipul *TABLE*; se declara tabloul indexat *PL/SQL* de acest tip.

Declararea tipului *TABLE* se face respectând urmatoarea sintaxa:

TYPE** nume_tip **IS TABLE OF

{*tip_coloana* | *variabila**%TYPE*** | *nume_tabel.coloana**%TYPE*** [***NOT NULL***] | *nume_tabel**%ROWTYPE***}

***INDEX BY** tip_indexare*;

Identificatorul *nume_tip* este numele noului tip definit care va fi specificat în declararea tabloului

PL/SQL, iar *tip_coloana* este un tip scalar simplu (de exemplu, *VARCHAR2*, *CHAR*, *DATE* sau *NUMBER*).

Pâna la versiunea *Oracle9i* unicul tip de indexare acceptat era *INDEX BY BINARY_INTEGER*. *Oracle9i* permite urmatoarele optiuni pentru *tip_indexare*: *PLS_INTEGER*, *NATURAL*, *POSITIVE*, *VARCHAR2*(*n*) sau chiar indexarea dupa un tip declarat cu *%TYPE*. Nu sunt permise indexarile *INDEX BY NUMBER*, *INDEX BY INTEGER*, *INDEX BY DATE*, *INDEX BY VARCHAR2*, *INDEX BY CHAR*(*n*) sau indexarea dupa un tip declarat cu *%TYPE* în care intervine unul dintre tipurile enumerate anterior.

***Observatii*:**

- Elementele unui tablou indexat nu sunt într-o ordine particulara si pot fi inserate cu chei arbitrare.
- Deoarece nu exista constrângeri de dimensiune, dimensiunea tabloului se modifica dinamic.
- Tabloul indexat *PL/SQL* nu trebuie initializat.
- Un tablou indexat neinitializat este vid.
- Un element al tabloului este nedefinit atâta timp cât nu are atribuita o valoare efectiva.
- Initial, un tablou indexat este nedens. Dupa declararea unui tablou se poate face referire la liniile lui prin precizarea valorii cheii primare.
- Daca se face referire la o linie care nu exista, atunci se produce exceptia *NO_DATA_FOUND*.
- Daca se doreste contorizarea numarului de linii, trebuie declarata o variabila în acest scop

sau poate fi utilizata o metoda asociata tabloului.

- Deoarece numarul de linii nu este limitat, operatia de adaugare de linii este restrictionata doar de dimensiunea memoriei alocate.
- Tablourile pot sa apara ca argumente într-o procedura.

Pentru inserarea unor valori din tablourile *PL/SQL* într-o coloana a unui tabel de date se utilizeaza instructiunea *INSERT* în cadrul unei secvente repetitive *LOOP*. Asemanator, pentru regasirea unor valori dintr-o coloana a unei baze de date într-un tablou *PL/SQL* se utilizeaza instructiunea *FETCH* (cursoare) sau instructiunea de atribuire în cadrul unei secvente repetitive *LOOP*.

Pentru a sterge liniile unui tablou fie se asigneaza elementelor tabloului valoarea *null*, fie se declara un alt tablou *PL/SQL* (de acelasi tip) care nu este initializat si acest tablou vid se asigneaza tabloului *PL/SQL* care trebuie sters. În *PL/SQL* 2.3 stergerea liniilor unui tabel se poate face utilizând metoda *DELETE*.

***Exemplu*:**

Sa se defineasca un tablou indexat *PL/SQL* având elemente de tipul *NUMBER*. Sa se introduca 20 de elemente în acest tablou. Sa se stearga tabloul.

```
DECLARE
TYPE tablou_numar IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
v_tablou tablou_numar; BEGIN
FOR i IN 1..20 LOOP
v_tablou(i) := i*i; DBMS_OUTPUT.PUT_LINE(v_tablou(i));
END LOOP;
--v_tablou := NULL;
--aceasta atribuire da eroarea PLS-00382 FOR i IN
v_tablou.FIRST..v_tablou.LAST LOOP
v_tablou(i) := NULL; END LOOP;
DBMS_OUTPUT.PUT_LINE('tabloul are ' || v_tablou.COUNT || '
elemente');
END;
```

Vectori

Vectorii (*varray*) sunt structuri asemanatoare vectorilor din limbajele *C* sau *Java*. Spre deosebire de tablourile indexate, vectorii au o dimensiune maxima (constanta) stabilita la declarare. În special, se utilizeaza pentru modelarea relatiilor *one-to-many*, atunci când numarul maxim de elemente din partea „*many*“ este cunoscut si ordinea elementelor este importanta.

Vectorii reprezinta structuri dense. Fiecare element are un index care da pozitia sa în vector si

care este folosit pentru accesarea elementelor particulare. Limita inferioara a indicelui este 1. Vectorul poate contine un numar variabil de elemente, de la 0 (vid) la numarul maxim specificat obligatoriu în definitia sa.

Tipul de date vector este declarat utilizând sintaxa:

TYPE *nume_tip* ***IS***

{***VARRAY*** | ***VARYING ARRAY***} (*lungime_maxima*)

OF *tip_elemente* [***NOT NULL***];

Identificatorul *nume_tip* este numele tipului de date vector, iar *lungime_maxima* reprezinta numarul maxim de elemente din vector. *Tip_elemente* este un tip scalar *PL/SQL*, tip înregistrare sau tip obiect. De asemenea, acest tip poate fi definit utilizând atributele *%TYPE* sau *%ROWTYPE*.

În *Oracle9i* sunt permise (pentru *tip_elemente*) tipurile *TABLE* sau alt tip *VARRAY*. Exista restrictii referitoare la tipul elementelor, în sensul ca acesta nu poate sa fie *BOOLEAN*, *NCHAR*, *NCLOB*, *NVARCHAR2*, *REF CURSOR*, *PLS_INTEGER*, *LONG*, *LONG RAW*, *NATURAL*, *NATURALN*, *POSITIVE*, *POSITIVEN*,

BINARY_INTEGER, *SIGNTYPE*, *STRING*, tip obiect cu atribute *TABLE* sau *VARRAY*, *BLOB*, *CLOB*, tip obiect cu atribute *BLOB* sau *CLOB.*

***Exemplu*:**

```
DECLARE
TYPE secventa IS VARRAY(5) OF VARCHAR2(10);
v_secsecventa := secventa ('alb','negru','rosu','verde'); BEGIN
v_sec (3) := 'rosu'; v_sec.EXTEND;
v_sec(5) := 'albastru';
-- extinderea la 6 elemente va genera eroarea ORA-06532
v_sec.EXTEND;
END;
```

Tablouri imbricate

Tablourile imbricate (*nested table*) sunt tablouri indexate a caror dimensiune nu este stabilita. Un tablou imbricat este o multime neordonata de elemente de acelasi tip. Valorile de acest tip pot fi stocate în baza de date, pot fi prelucrate direct în instructiuni *SQL* si au exceptii predefinite proprii. Numarul maxim de linii ale unui tablou imbricat este dat de capacitatea maxima 2 *GB.*

Sistemul *Oracle* nu stocheaza liniile unui tablou imbricat într-o ordine particulara. Dar, când se regaseste tabloul în variabile *PL/SQL*, liniile vor avea indici consecutivi începând cu valoarea 1. Initial, aceste tablouri sunt structuri dense, dar se poate ca în urma prelucrarii sa nu mai aiba indici consecutivi.

Comanda de declarare a tipului de date tablou imbricat are sintaxa:

TYPE *nume_tip* ***IS TABLE OF*** *tip_ elemente* [***NOT NULL***];

Identificatorul *nume_tip* reprezinta numele noului tip de date tablou imbricat, iar *tip_elemente* este tipul fiecarui element din tabloul imbricat, care poate fi un tip definit de utilizator sau o expresie cu *%TYPE*, respective *%ROWTYPE*.

În *Oracle9i* sunt permise (pentru *tip_elemente*) tipurile *TABLE* sau alt tip *VARRAY*. Exista restrictii referitoare la tipul elementelor, în sensul ca acesta nu poate sa fie *BOOLEAN, STRING, NCHAR, NCLOB, NVARCHAR2, REF CURSOR, BINARY_INTEGER, PLS_INTEGER, LONG, LONG RAW, NATURAL,*

NATURALN, POSITIVE, POSITIVEN, SIGNTYPE, tip obiect cu atributele *TABLE* sau *VARRAY*.

Tabloul imbricat are o singura coloana, iar daca aceasta este de tip obiect, tabloul poate fi vizualizat ca un tabel multicoloana, având câte o coloana pentru fiecare atribut al tipului obiect.

***Exemplu*:**

```
DECLARE
TYPE numartab IS TABLE OF NUMBER;
-- se creeaza un tablou cu un singur element v_tab_1  numartab := numartab(-7);
-- se creeaza un tablou cu 4 elemente v_tab_2    numartab := numartab(7,9,4,5);
-- se creeaza un tablou fara nici un element v_tab_3  numartab := numartab();
BEGIN
v_tab_1(1) := 57; FOR j IN 1..4 LOOP
DBMS_OUTPUT.PUT_LINE (v_tab_2(j) || ' '); END LOOP;
END;
```

Se observa ca singura diferenta sintactica între tablourile indexate si cele imbricate este absenta clauzei

INDEX BY BINARY_INTEGER. Mai exact, daca aceasta clauza lipseste, tipul este tablou imbricat.

***Observatii*:**

- Spre deosebire de tablourile indexate, vectorii si tablourile imbricate pot sa apara în definirea tabelelor bazei de date.
- Tablourile indexate pot avea indice negativ, domeniul permis pentru index fiind – 2147483647..2147483647, iar pentru tabele imbricate domeniul indexului este 1..2147483647.
- Tablourile imbricate, spre deosebire de tablourile indexate, pot fi prelucrate prin comenzi *SQL*.

- Tablourile imbricate trebuie initializate si/sau extinse pentru a li se adauga elemente.

Când este creat un tablou indexat care nu are înca elemente, el este vid. Daca un tablou imbricat (sau un vector) este declarat, dar nu are înca nici un element (nu este initializat), el este automat initializat (atomic) *null*. Adica, colectia este *null*, nu elementele sale. Prin urmare, pentru tablouri imbricate poate fi utilizat operatorul *IS NULL*. Daca se încearca sa se adauge un element la un tablou imbricat *null*, se va genera eroarea „*ORA - 06531: reference to uninitialized collection*“ care corespunde exceptiei predefinite *COLLECTION_IS_NULL*.

Prin urmare, cum poate fi initializat un tablou imbricat? Ca si obiectele, vectorii si tablourile imbricate sunt initializate cu ajutorul constructorului. Acesta are acelasi nume ca si tipul colectiei referite. *PL/SQL* apeleaza un constructor numai în mod explicit. Tabelele indexate nu au constructori.

Constructorul primeste ca argumente o lista de valori de tip *tip_elemente*. Elementele sunt numerotate în ordine, de la 1 la numarul de valori date ca parametrii constructorului. Dimensiunea initiala a colectiei este egala cu numarul de argumente date în constructor, când aceasta este initializata. Pentru vectori nu poate fi depasita dimensiunea maxima precizata la declarare. Atunci când constructorul este fara argumente, va crea un obiect fara nici un element, dar care nu are valoarea *null*. Exemplul urmator este concludent în acest sens.

***Exemplu*:**

```
DECLARE
TYPE alfa IS TABLE OF VARCHAR2(50);
-- creeaza un tablou null tab1    alfa ;
-- creeaza un tablou cu un element care este null tab2 alfa :=
alfa() ;
BEGIN
IF tab1 IS NULL THEN
DBMS_OUTPUT.PUT_LINE('tab1 este NULL'); ELSE
DBMS_OUTPUT.PUT_LINE('tab1 este NOT NULL'); END IF;
IF tab2 IS NULL THEN DBMS_OUTPUT.PUT_LINE('tab2 este NULL');
ELSE
DBMS_OUTPUT.PUT_LINE('tab2 este NOT NULL'); END IF;
END;
```

În urma executiei acestui bloc se obtine urmatorul rezultat:

```
tab1 este NULL tab2 este NOT NULL
```

Exceptiile semnificative care apar în cazul utilizarii incorecte a colectiilor sunt prezentate în exemplul care urmeaza.

Exemplu:

```
DECLARE
TYPE numar IS TABLE OF INTEGER;
alfa numar;
BEGIN
alfa(1) := 77;
-- declanseaza exceptia COLLECTION_IS_NULL alfa := numar(15,
26, 37);
alfa(1) := ASCII('X');
alfa(2) := 10*alfa(1); alfa('P') := 77;
/* declanseaza exceptia VALUE_ERROR deoarece indicele nu este
convertibil la intreg */
alfa(4) := 47;
/* declanseaza exceptia SUBSCRIPT_BEYOND_COUNT deoarece
indicele se refera la un element neinitializat */
alfa(null) := 7; -- declanseaza exceptia VALUE_ERROR alfa(0) :=
7; -- exceptia SUBSCRIPT_OUTSIDE_LIMIT alfa.DELETE(1);
IF alfa(1) = 1 THEN … -- exceptia NO_DATA_FOUND
… END;
```

Tablourile imbricate si vectorii pot fi utilizati drept câmpuri în tabelele bazei. Aceasta presupune ca fiecare înregistrare din tabelul respectiv contine un obiect de tip colectie. Înainte de utilizare, tipul trebuie stocat în dictionarul datelor, deci trebuie declarat prin comanda:

CREATE TYPE *nume_tip* ***AS*** {***TABLE*** | ***VARRAY***} ***OF*** *tip_elemente*;

Dupa crearea tabelului (prin comanda *CREATE TABLE*), pentru fiecare câmp de tip tablou imbricat din tabel este necesara clauza de stocare:

NESTED TABLE *nume_câmp* ***STORE AS*** *nume_tabel*;

Colectii pe mai multe niveluri

În *Oracle9i* se pot construi colectii pe mai multe niveluri (*multilevel collections*), prin urmare colectii ale caror elemente sunt, în mod direct sau indirect, colectii. În felul acesta pot fi definite structuri complexe: vectori de vectori, vectori de tablouri imbricate, tablou imbricat de vectori, tablou imbricat de tablouri imbricate, tablou imbricat sau vector de un tip definit de utilizator care are un atribut de tip tablou imbricat sau vector.

Aceste structuri complexe pot fi utilizate ca tipuri de date pentru definirea coloanelor unui tabel relational, ale atributelor unui obiect într-un tabel obiect sau ale variabilelor *PL/SQL*. Numarul nivelurilor de imbricare este limitat doar de capacitatea de stocare a sistemului.

Pentru a accesa un element al colectiei incluse sunt utilizate doua seturi de paranteze. Obiectele de tipul colectie pe mai multe niveluri nu pot fi comparate.

***Exemplu*:**

În exemplele care urmeaza sunt definite trei structuri complexe si sunt prezentate câteva modalitati de utilizare ale acestora. Exemplele se refera la vectori pe mai multe niveluri, tablouri imbricate pe mai multe niveluri si tablouri indexate pe mai multe niveluri.

```
DECLARE
TYPEalfa IS VARRAY(10) OF INTEGER; TYPEbeta IS VARRAY(10) OF
alfa;
valfalfa := alfa(12,31,5); --initializare
vbetbeta := beta(valf,alfa(55,6,77),alfa(2,4),valf); i
    integer;
var1alfa;
BEGIN
i := vbet(2)(3); -- i va lua valoarea 77
vbet.EXTEND; -- se adauga un element de tip vector la vbet
vbet(5) := alfa(56,33);
vbet(4) := alfa(44,66,77,4321);
vbet(4)(4) := 7; -- 4321 este inlocuit cu 7
vbet(4).EXTEND; -- se adauga un element la al 4-lea element
vbet(4)(5) := 777; -- acest nou element adaugat va fi 777
END;
/

DECLARE
TYPEgama IS TABLE OF VARCHAR2(20);
TYPEdelta IS TABLE OF gama;
TYPEteta IS VARRAY(10) OF INTEGER;
TYPEepsi IS TABLE OF teta;
var1gama := gama('alb','negru'); var2  delta := delta(var1);
var3epsi := epsi(teta(31,15),teta(1,3,5)); BEGIN
var2.EXTEND;
var2(2) := var2(1);
var2.DELETE(1); -- sterge primul element din var2
/* sterge primul sir de caractere din al doilea tabel al
tabelului imbricat */
```

```
var2(2).DELETE(1); END;
/ DECLARE
TYPE alfa IS TABLE OF INTEGER INDEX BY BINARY_INTEGER; TYPE beta
IS TABLE OF alfa INDEX BY BINARY_INTEGER; TYPE   gama IS VARRAY(10)
OF VARCHAR2(30);
TYPE delt IS TABLE OF gama INDEX BY BINARY_INTEGER;
var1 gama := gama('alb','negru');
var2 beta; var3 delt; var4  alfa;
var5 alfa; -- tabel null BEGIN
var4(1) := 324;
var4(2) := 222;
var4(42) := 333;
var2(27) := var4;
var3(39) := gama(77,76,89,908);
-- var2(40)(3) := 55; eroare nu exista elementul 40 in var2
var2(40) := var5; -- asignez un tabel null
var2(40)(3) := 55; -- corect END;
/
```

Prelucrarea colectiilor

O colectie poate fi exploatata fie în întregime (atomic) utilizând comenzi *LMD*, fie pot fi prelucrate elemente individuale dintr-o colectie (*piecewise updates*) utilizând operatori *SQL* sau anumite facilitati oferite de *PL/SQL*.

Comanda *INSERT* permite inserarea unei colectii într-o linie a unui tabel. Colectia trebuie sa fie creata si initializata anterior. Comanda *UPDATE* este folosita pentru modificarea unei colectii stocate, iar *DELETE* poate sterge o linie ce contine o colectie. Colectiile din baza de date pot fi regasite în variabile *PL/SQL*, utilizând comanda *SELECT*.

***Exemplu*:**

```
CREATE OR REPLACE TYPE operalist AS VARRAY(10) OF NUMBER(4);
CREATE TABLE gal_ope ( cod_galerie NUMBER(10), nume_galerie
            VARCHAR2(20), info    operalist);
DECLARE
v_opera  operalist := operalist (777, 888, 999); v_info_op
         operalist := operalist (7007);
v_info   gal_ope.info%TYPE;
v_cod    gal_ope.cod_galerie%TYPE; BEGIN
```

```
     INSERT INTO gal_ope
     VALUES (4567, 'Impresionisti', operalist(4567,4987)); INSERT
INTO gal_ope
     VALUES (2345, 'Cubism', v_opera); INSERT INTO gal_ope
     VALUES (123, 'Alfa', v_info_op); SELECT info
     INTO v_info
     FROM gal_ope
     WHERE cod_galerie = v_cod; END;
```

Un vector stocat într-un tabel este prelucrat ca un întreg (nu pot fi modificate elemente individuale). Prin urmare, elementele individuale ale unui vector nu pot fi referite în comenzile *INSERT*, *UPDATE* sau *DELETE*. Pentru referirea acestora trebuie utilizate comenzi procedurale *PL/SQL*. Pentru a modifica un vector, el trebuie selectat într-o variabila *PL/SQL* a carei valoare poate fi modificata si apoi reinserata în tabel.

Tablourile imbricate depuse în baza de date sunt mai flexibile, deoarece pot fi prelucrate fie în întregime, fie ca elemente individuale. În fiecare caz pot fi utilizate numai comenzi *SQL*. Se pot face reactualizari sau inserari asupra tablourilor imbricate care dau o valoare noua pentru întreaga colectie sau se pot face inserari, stergeri, reactualizari de elemente particulare din colectie.

O colectie poate fi asignata altei colectii prin comenzile *INSERT*, *UPDATE*, *FETCH*, *SELECT*, instructiunea de atribuire sau prin apelul unui subprogram, dar colectiile trebuie sa fie de acelasi tip. Daca unei colectii i se asigneaza o colectie atomic *null*, aceasta devine atomic *null* si trebuie reinitializata.

În *Oracle8i* a fost introdus operatorul *TABLE*, ce permite prelucrarea elementelor unui tablou imbricat care este stocat într-un tabel. Operatorul permite interogarea unei colectii în clauza *FROM* (la fel ca un tabel).

Operandul lui *TABLE* este fie numele unei colectii si atunci rezultatul operandului este tot o colectie, fie este o subinterogare referitoare la o colectie, iar în acest caz, operatorul *TABLE* returneaza o singura valoare (coloana) care este un tablou imbricat sau un vector. Prin urmare, lista din clauza *SELECT* a subcererii trebuie sa aiba un singur articol. Subcererea nu poate returna colectii pentru mai multe linii.

***Exemplu*:**

Se presupune ca tabelul *opera* are o coloana *info* de tip tablou imbricat. Acest tablou are doua componente în care pentru fiecare opera de arta sunt depuse numele articolului referitor la opera respectiva si revista în care a aparut. Sa se insereze o linie în tabelul imbricat.

```
     INSERT INTO TABLE (SELECT info
     FROM                    opera
     WHERE                   titlu = 'Primavara') VALUES ('Pictura
```

```
moderna', 'Orizonturi');
```

Listarea codului fiecarei opere de arta si a colectiei articolelor referitoare la aceste opere de arta se face prin comanda:

```
SELECT    a.cod_opera, b.*
FROMopera a, TABLE (a.info) b;
```

Pentru tablouri imbricate pe mai multe niveluri, operatiile *LMD* pot fi facute atomic sau pe elemente individuale, iar pentru vectori pe mai multe niveluri, operatiile pot fi facute numai atomic.

Pentru prelucrarea unei colectii locale se folosesc operatorii *TABLE* si *CAST*. *CAST* are forma sintactica:

CAST (*nume_colectie* ***AS*** *tip_colectie*)

Operanzii lui *CAST* sunt o colectie declarata local (de exemplu, într-un bloc *PL/SQL* anonim) si un tip colectie *SQL*. *CAST* converteste colectia locala la tipul specificat. În felul acesta, o colectie poate fi prelucrata ca si cum ar fi un tabel *SQL* al bazei de date.

Metodele unei colectii

PL/SQL ofera subprograme numite metode (*methods*), care opereaza asupra unei colectii. Acestea pot fi apelate numai din comenzi procedurale, si nu din *SQL*.

Metodele sunt apelate prin expresia:

nume_colectie.*nume_metoda* [(*parametri*)]

Metodele care se pot aplica colectiilor *PL/SQL* sunt urmatoarele:

- *COUNT* returneaza numarul curent de elemente ale unei colectii *PL/SQL*;
- *DELETE*(*n*) sterge elementul *n* dintr-o colectie *PL/SQL*; *DELETE*(*m*, *n*) sterge toate elementele având indecsii între *m* si *n*; *DELETE* sterge toate elementele unei colectii *PL/SQL* (nu este valida pentru tipul *varrays*);
- *EXISTS*(*n*) returneaza *TRUE* daca exista al *n*-lea element al unei colectii *PL/SQL* (altfel, returneaza

FALSE, chiar daca elementul este *null*);

- *FIRST*, *LAST* returneaza indicele primului, respectiv ultimului element din colectie;
- *NEXT*(*n*), *PRIOR*(*n*) returneaza indicele elementului urmator, respectiv precedent celui de rang *n* din colectie, iar daca nu exista un astfel de element returneaza valoarea *null*;
- *EXTEND* adauga elemente la sfârsitul unei colectii: *EXTEND* adauga un element *null* la sfârsitul colectiei, *EXTEND*(*n*) adauga *n* elemente *null*, *EXTEND*(*n*, *i*) adauga *n* copii ale elementului de rang *i* (nu este valida pentru tipul *index-by tables*);
- *LIMIT* returneaza numarul maxim de elemente ale unei colectii (cel de la declarare) pentru tipul vector si *null* pentru tablouri imbricate (nu este valida pentru tipul *index-by tables*);

- *TRIM* sterge elementele de la sfârsitul unei colectii: *TRIM* sterge ultimul element, *TRIM*(*n*) sterge ultimele *n* elemente (nu este valida pentru tipul *index-by tables*). Similar metodei *EXTEND*, metoda *TRIM* opereaza asupra dimensiunii interne a tabloului imbricat.

EXISTS este singura metoda care poate fi aplicata unei colectii atomice *null*. Orice alta metoda declanseaza exceptia *COLLECTION_IS_NULL*.

COUNT, *EXISTS*, *FIRST*, *LAST*, *NEXT*, *PRIOR* si *LIMIT* sunt functii, iar restul sunt proceduri *PL/SQL*.

Bulk bind

În exemplul care urmeaza, comanda *DELETE* este trimisa motorului *SQL* pentru fiecare iteratie a comenzii *FOR*.

***Exemplu*:**

```
DECLARE
TYPE nume IS VARRAY(20) OF NUMBER;
alfa nume := nume(10,20,70); -- coduri ale galeriilor BEGIN
FOR j IN alfa.FIRST..alfa.LAST LOOP
DELETE FROM opera
WHERE  cod_galerie = alfa (j); END LOOP;
END;
```

Pentru a realiza mai rapid aceasta operatie, ar trebui sa existe posibilitatea de a sterge (prelucra) întreaga colectie si nu elemente individuale. Mecanismul care permite acest lucru este cunoscut sub numele *bulk bind.*

În timpul compilarii, motorul *PL/SQL* asociaza identificatoriilor o adresa, un tip de date si o valoare.

Acest proces este numit *binding*.

Comenzile *SQL* din blocurile *PL/SQL* sunt trimise motorului *SQL* pentru a fi executate. Motorul *SQL* poate trimite date înapoi motorului *PL/SQL* (de exemplu, ca rezultat al unei interogari). De multe ori, datele care trebuie prelucrate apartin unei colectii, iar colectia este iterata printr-un ciclu *FOR*. Prin urmare, transferul (în ambele sensuri) între *SQL* si *PL/SQL* are loc pentru fiecare linie a colectiei.

Începând cu *Oracle8i* exista posibilitatea ca toate liniile unei colectii sa fie transferate simultan printr-o singura operatie. Procedeul este numit *bulk bind* si este realizat cu ajutorul comenzii *FORALL*, ce poate fi folosita cu orice tip de colectie.

Comanda *FORALL* are sintaxa:

FORALL *index* ***IN*** *lim_inf..lim_sup comanda_sql*;

Motorul *SQL* executa *comanda_sql* o singura data pentru toate valorile indexului. *Comanda_sql*

este una dintre instructiunile *INSERT*, *UPDATE*, *DELETE* care refera elementele uneia sau mai multor colectii. Variabila *index* poate fi referita numai în comanda *FORALL* si numai ca indice de colectie.

Exemplul care urmeaza optimizeaza problema anterioara, în sensul ca *DELETE* este trimisa motorului

SQL o singura data, pentru toate liniile colectiei.

***Exemplu*:**

```
DECLARE TYPE nume IS VARRAY(20) OF NUMBER;
alfa nume := nume(10,20,70); -- coduri ale galeriilor BEGIN
…
FORALL j IN alfa.FIRST..alfa.LAST DELETE FROM opera
                  WHERE cod_galerie = alfa (j);
                                END;
```

Pentru utilizarea comenzii *FORALL* sunt necesare urmatoarele restrictii:

- comanda poate fi folosita numai în programe *server-side*, altfel apare eroarea „*this feature is not supported in client-side programs*“;
- comenziile *INSERT, UPDATE, DELETE* trebuie sa refere cel putin o colectie;
- toate elementele colectiei din domeniul precizat trebuie sa existe (daca, de exemplu, un element a fost sters, atunci este semnalata o eroare);
- indicii colectiilor nu pot sa fie expresii si trebuie sa aiba valori continue.

Daca exista o eroare în procesarea unei linii printr-o operatie *LMD* de tip *bulk*, numai acea linie va fi derulata înapoi (*rollback*).

Cursorul *SQL* are un atribut compus, *%BULK_ROWCOUNT*, care numara liniile afectate de iteratiile comenzii *FORALL*. *%BULK_ROWCOUNT*(*i*) reprezinta numarul de linii procesate de a *i*-a executie a comenzii *SQL*. Atributul nu poate fi parametru într-un subprogram si nu poate fi asignat altei colectii.

Începând cu *Oracle9i*, este inclusa o noua clauza în comanda *FORALL*. Clauza, numita *SAVE EXCEPTIONS*, permite ca toate exceptiile care apar în timpul executiei comenzii *FORALL* sa fie salvate si astfel procesarea poate sa continue.

În acest context, poate fi utilizat atributul cursor *%BULK_EXCEPTIONS* pentru a vizualiza informatii despre aceste exceptii. Atributul actioneaza ca un tablou *PL/SQL* si are doua câmpuri:

- *%BULK_EXCEPTIONS*(*i*).*ERROR_INDEX*, reprezentând iteratia în timpul careia s-a declansat exceptia;
- *%BULK_EXCEPTIONS*(*i*).*ERROR_CODE*, reprezentând codul *Oracle* al erorii respective.

Regasirea rezultatului unei interogari în colectii (înainte de a fi trimisa motorului *PL/SQL*) se

poate obtine cu ajutorul clauzei *BULK COLLECT*.

Aceasta clauza poate sa apara în comenzile *SELECT INTO* (cursoare implicite*), FETCH INTO* (cursoare explicite) sau în clauza *RETURNING INTO* a comenzilor *INSERT*, *UPDATE*, *DELETE*. În cazul cursoarelor explicite, numarul liniilor încarcate din baza de date poate fi limitat utilizând optiunea *LIMIT*.

Clauza *BULK COLLECT* are urmatoarea sintaxa:

BULK COLLECT INTO *nume_colectie* [, *nume_colectie…*]

***Exemplu*:**

```
DECLARE
TYPE tip1 IS TABLE OF opera.cod_opera%TYPE; TYPE tip2 IS TABLE
OF opera.titlu%TYPE; alfa    tip1;
beta tip2;
BEGIN
…
/* motorul SQL incarca in intregime coloanele cod_opera si
titlu in tabelele imbricate, inainte de a returna tabelele motorului
PL/SQL */
SELECT cod_opera, titlu BULK COLLECT INTO alfa,beta FROM opera;
…
/* daca exista n opere de arta in stare buna, atunci alfa va
contine codurile celor n opere de arta */
DELETE FROM opera WHERE stare = 'buna' RETURNING cod_opera BULK
COLLECT INTO alfa;
… END;
```

Comanda *FORALL* se poate combina cu clauza *BULK COLLECT*. Totusi, trebuie subliniat ca ele nu pot fi folosite simultan în comanda *SELECT*.