

Tipuri de date compuse

- înregistrare (**RECORD**) ;
- colecție (**INDEX-BY TABLE, NESTED TABLE, VARRAY**).

I. Înregistrări (RECORD)

- Declararea tipului **RECORD** se face conform următoarei sintaxe:

```
TYPE nume_tip IS RECORD  
    (nume_câmp1 {tip_câmp | variabilă%TYPE |  
      nume_tabel.coloană%TYPE | nume_tabel%ROWTYPE}  
    [ [NOT NULL] {:= | DEFAULT} expresie1],  
    (nume_câmp2 {tip_câmp | variabilă%TYPE |  
      nume_tabel.coloană%TYPE | nume_tabel%ROWTYPE}  
    [ [NOT NULL] {:= | DEFAULT} expresie2],...);
```

- Oracle9i introduce câteva facilități legate de acest tip de date.
 - Se poate insera (**INSERT**) o linie într-un tabel utilizând tipul **RECORD**.
 - Se poate actualiza (**UPDATE**) o linie într-un tabel utilizând tipul **RECORD** (cu sintaxa **SET ROW**)
 - se poate regăsi și returna sau șterge informația din clauza **RETURNING** a comenzilor **UPDATE** sau **DELETE**.
 - dacă în comenzile **UPDATE** sau **DELETE** se modifică mai multe linii, atunci pot fi utilizate în sintaxa **BULK COLLECT INTO**, colecții de înregistrări.

Exerciții:

1. Să se șteargă angajatul având codul 200 din tabelul **EMP**. Să se rețină într-o variabilă de tip **RECORD** codul, numele, salariul și departamentul acestui angajat (clauza **RETURNING**) . Să se afișeze înregistrarea respectivă. Rollback.

2. a) Folosind tipul declarat mai sus, să se adauge o linie în tabelul **EMP** prin intermediul unei variabile de tip înregistrare inițializate. Efectuați modificările necesare asupra tipului de date, astfel încât această inserare să fie posibilă. La inițializarea unei variabile de tip record, țineți cont de constrângerile **NOT NULL** definite asupra tabelului **EMP**.

b) Modificați valoarea unei componente a variabilei definite anterior și actualizați conținutul liniei (**SET ROW**) introduse în tabel.

II. Colecții

Colecțiile permit să fie prelucrate simultan mai multe variabile de același tip. Fiecare element are un indice unic, care determină poziția sa în colecție.

În **PL/SQL** există trei tipuri de colecții:

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

Obs :

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

- Singura diferență sintactică între tablourile indexate și cele imbricate este clauza *INDEX BY*. Dacă această clauză lipsește, atunci tipul este tablou imbricat.

➤ Atribute și metode ale unei colecții: (informații complete – în curs !)

Atribut sau metodă	Descriere
COUNT	numărul componentelor colecției
FIRST	Indicele primului element din tablou
LAST	Indicele ultimului element din tablou
EXISTS	întoarce TRUE dacă există în tablou componenta cu indexul specificat
NEXT	returnează indicele următoarei componente
PRIOR	returnează indicele componentei anterioare
DELETE	șterge una sau mai multe componente.
EXTEND	Adaugă elemente la sfârșit
LIMIT	Numărul maxim de elemente al unei colecții (pentru vectori), null pentru tablouri imbricate
TRIM	șterge elementele de la sfârșitul unei colecții

Ultimele 3 metode nu sunt valide pentru index-by tables.

➤ **bulk bind** permite ca toate liniile unei colecții să fie transferate simultan printr-o singură operație.

• este realizat cu ajutorul comenzii *FORALL*, ce poate fi folosită cu orice tip de colecție:

FORALL index *IN* lim_inf..lim_sup
comanda_sql;

Cursorul SQL are un atribut compus *%BULK_ROWCOUNT* care numără liniile afectate de iterațiile comenzii *FORALL*. *%BULK_ROWCOUNT(i)* reprezintă numărul de linii procesate de a *i*-a execuție a comenzii SQL.

➤ **Regăsirea rezultatului unei interogări în colecții** (înainte de a fi trimisă motorului *PL/SQL*) se poate obține cu ajutorul clauzei *BULK COLLECT*:

...BULK COLLECT INTO nume_colecție [,nume_colecție]...

➤ Clauza poate să apară în:

- comenzile *SELECT INTO* (cursoare implicite),
- comenzile *FETCH INTO* (cursoare explicite),
- clauza *RETURNING INTO* a comenzilor *INSERT*, *UPDATE*, *DELETE*.

Exerciții:

3. Analizați și comentați exemplul următor. Afișați valorile variabilelor definite.

```
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;
  i INTEGER;
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);
  -- afisati valorile variabilelor definite; exemplu dat pentru v_tab_imbri
  i:=v_tab_imbri.FIRST;
  WHILE (i <= v_tab_imbri.LAST) LOOP
    DBMS_OUTPUT.PUT_LINE('v_tab_imbri: '||v_tab_imbri(i));
    i:= v_tab_imbri.NEXT(i);
  END LOOP;
END;
```

/

II.1. Tablouri indexate (index-by tables)

➤ Tabloul indexat *PL/SQL* are două componente:

- coloană ce cuprinde cheia primară pentru acces la liniile tabloului
- o coloană care include valoarea efectivă a elementelor tabloului.

➤ Declararea tipului *TABLE* se face respectând următoarea sintaxă:

```
TYPE nume_tip IS TABLE OF
    {tip_coloană | variabilă%TYPE |
    nume_tabel.coloană%TYPE [NOT NULL] |
    nume_tabel%ROWTYPE}
INDEX BY tip_indexare;
```

Observații:

- Elementele unui tablou indexat nu sunt într-o ordine particulară și pot fi inserate cu chei arbitrare.
 - Deoarece nu există constrângeri de dimensiune, dimensiunea tabloului se modifică dinamic.
 - Tabloul indexat *PL/SQL* nu poate fi inițializat în declararea sa.
 - Un tablou indexat neinițializat este vid (nu conține nici valori, nici chei).
 - Un element al tabloului este nedefinit atâta timp cât nu are atribuită o valoare efectivă.
 - Dacă se face referire la o linie care nu există, atunci se produce excepția *NO_DATA_FOUND*.
- Pentru inserarea unor valori din tablourile *PL/SQL* într-o coloană a unui tabel de date se utilizează instrucțiunea *INSERT* în cadrul unei secvențe repetitive *LOOP*.
- Pentru regăsirea unor valori dintr-o coloană a unei baze de date într-un tablou *PL/SQL* se utilizează instrucțiunea *FETCH* (cursoare) sau instrucțiunea de atribuire în cadrul unei secvențe repetitive *LOOP*.
- Pentru a șterge liniile unui tablou fie se asignează elementelor tabloului valoarea *null*, fie se declară un alt tablou *PL/SQL* (de același tip) care nu este inițializat și acest tablou vid se

asignează tabloului *PL/SQL* care trebuie șters. În *PL/SQL 2.3* ștergerea liniilor unui tabel se poate face utilizând metoda *DELETE*.

Exerciții:

4. Să se definească un tablou indexat *PL/SQL* având elemente de tipul *NUMBER*. Să se introducă 20 de elemente în acest tablou. Să se afișeze, apoi să se șteargă tabloul utilizând diverse metode.
5. Să se definească un tablou de înregistrări având tipul celor din tabelul *dept*. Să se inițializeze un element al tabloului și să se introducă în tabelul *dept*. Să se șteargă elementele tabloului.

II.2 Vectori (*varray*)

- Vectorii (*varray*) sunt structuri asemănătoare vectorilor din limbajele C sau Java.
- Vectorii au o dimensiune maximă (constantă) stabilită la declarare. În special, se utilizează pentru modelarea relațiilor *one-to-many*, atunci când numărul maxim de elemente din partea „many” este cunoscut și ordinea elementelor este importantă.
- Fiecare element are un index, a cărui limită inferioară este 1.

➤ Tipul de date vector este declarat utilizând sintaxa:

```
TYPE nume_tip IS
    {VARRAY | VARYING ARRAY} (lungime_maximă)
    OF tip_elemente [NOT NULL];
```

Exerciții:

6. Analizați și comentați exemplul următor.

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

Obs : Pentru a putea reține și utiliza tablourile imbricate și vectorii, trebuie să declarăm în SQL tipuri de date care să îi reprezinte.

Tablourile imbricate și vectorii pot fi utilizați drept câmpuri în tabelele bazei. Aceasta presupune că fiecare înregistrare din tabelul respectiv conține un obiect de tip colecție. Înainte de utilizare, tipul trebuie stocat în dicționarul datelor, deci trebuie declarat prin comanda:

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

7. a) Să se declare un tip *proiect* care poate reține maxim 50 de valori de tip *VARCHAR2(15)*.
 b) Să se creeze un tabel *test* având o coloană *cod_ang* de tip *NUMBER(4)* și o coloană *proiecte_alocate* de tip *proiect*. Ce relație se modelează în acest fel?
 c) Să se creeze un bloc *PL/SQL* care declară o variabilă (un vector) de tip *proiect*, introduce valori în aceasta iar apoi valoarea vectorului respectiv este introdusă pe una din liniile tabelului *test*.
8. Să se scrie un bloc care mărește salariile angajaților din departamentul 50 cu 10%, în cazul în care salariul este mai mic decât 5000. Se va utiliza un vector corespunzător codurilor angajaților.

Obs: Prin comanda *FORALL* sunt trimise toate datele pe server, executându-se apoi o singură comandă *SELECT*, *UPDATE* etc.

II.3 Tablouri imbricate

- Tablourile imbricate (*nested table*) sunt tablouri indexate a căror dimensiune nu este stabilită.
 - folosesc drept indici numere consecutive ;
 - sunt asemenea unor tabele cu o singură coloană;
 - nu au dimensiune limitată, ele cresc dinamic;
 - inițial, un tablou imbricat este dens (are elementele pe poziții consecutive) dar pot apărea spații goale prin ștergere ;
 - metoda NEXT ne permite să ajungem la următorul element ;
 - pentru a insera un element nou, tabloul trebuie extins cu metoda EXTEND(nr_comp) ;
- Un tablou imbricat este o mulțime neordonată de elemente de același tip. Valorile de acest tip:
 - pot fi stocate în baza de date,
 - pot fi prelucrate direct în instrucțiuni SQL
 - au excepții predefinite proprii.
- Comanda de declarare a tipului de date tablou imbricat are sintaxa:

TYPE nume_tip IS TABLE OF tip_ elemente [NOT NULL];
- Pentru adaugarea de linii într-un tablou imbricat, acesta trebuie să fie inițializat cu ajutorul constructorului.
 - PL/SQL apelează un constructor numai în mod explicit.
 - Tabelele indexate nu au constructori.
 - Constructorul primește ca argumente o listă de valori numerotate în ordine, de la 1 la numărul de valori date ca parametrii constructorului.
 - Dimensiunea inițială a colecției este egală cu numărul de argumente date în constructor, când aceasta este inițializată.
 - Pentru vectori nu poate fi depășită dimensiunea maximă precizată la declarare.
 - Atunci când constructorul este fără argumente, va crea o colecție fără nici un element (vida), dar care are valoarea *not null*.

Exerciții:

9. Să se declare un tip tablou imbricat și o variabilă de acest tip. Inițializați variabila și afișați conținutul tabloului, de la primul la ultimul element și invers.

DECLARE

TYPE CharTab IS TABLE OF CHAR(1);

v_Characters CharTab :=

CharTab('M', 'a', 'd', 'a', 'm', ',', ', ',
'l', '', 'm', ' ', 'A', 'd', 'a', 'm');

v_Index INTEGER;

BEGIN

v_Index := v_Characters.FIRST;

WHILE v_Index <= v_Characters.LAST LOOP

DBMS_OUTPUT.PUT(v_Characters(v_Index));

v_Index := v_Characters.NEXT(v_Index);

END LOOP;

DBMS_OUTPUT.NEW_LINE;

v_Index := v_Characters.LAST;

WHILE v_Index >= v_Characters.FIRST LOOP

DBMS_OUTPUT.PUT(v_Characters(v_Index));

v_Index := v_Characters.PRIOR(v_Index);

END LOOP;

DBMS_OUTPUT.NEW_LINE;

END;

/

10. Creați un tip tablou imbricat, numit NumTab. Afișați conținutul acestuia, utilizând metoda EXISTS. Atribuiți valorile tabloului unui tablou index-by. Afișați și acest tablou, în ordine inversă.

11. Să se analizeze următorul bloc PL/SQL. Ce se obține în urma execuției acestuia ?

```
DECLARE
  TYPE alfa IS TABLE OF VARCHAR2(50);
  -- creeaza un tablou (atomic) null
  tab1 alfa ;
  /* creeaza un tablou cu un element care este null, dar
    tabloul nu este null, el este initializat, poate
    primi elemente */
  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;
```

12. Analizați următorul exemplu, urmărind excepțiile semnificative care apar în cazul utilizării incorecte a colecțiilor:

```
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;
```

II.4 Colecții pe mai multe niveluri

!!! De analizat exemplele din curs!

II.5 Prelucrarea colecțiilor

- *INSERT* - permite inserarea unei colecții într-o linie a unui tabel. Colecția trebuie să fie creată și inițializată anterior.
- *UPDATE* este folosită pentru modificarea unei colecții stocate.
- *DELETE* poate șterge o linie ce conține o colecție.
- Colecțiile din baza de date pot fi regăsite în variabile *PL/SQL*, utilizând comanda *SELECT*.
- operatorul *TABLE* permite prelucrarea elementelor unui tablou imbricat care este stocat într-un tabel. Operatorul permite interogarea unei colecții în clauza *FROM* (la fel ca un tabel).
- Pentru tablouri imbricate pe mai multe niveluri, operațiile *LMD* pot fi făcute atomic sau pe elemente individuale, iar pentru vectori pe mai multe niveluri, operațiile pot fi făcute numai atomic.
- Pentru prelucrarea unei colecții locale se poate folosi și operatorul *CAST*. *CAST* are forma sintactică:

CAST (*nume_colecție AS tip_colecție*)

Exerciții:

13. a) Să se creeze un tip *LIST_ANG*, de tip vector, cu maxim 10 componente de tip *NUMBER(4)*.
 b) Să se creeze un tabel *JOB_EMP*, având coloanele: *cod_job* de tip *NUMBER(3)*, *titlu_job* de tip *VARCHAR2(25)* și *info* de tip *LIST_ANG*.
 c) Să se creeze un bloc *PL/SQL* care declară și inițializează două variabile de tip *LIST_ANG*, o variabilă de tipul coloanei *info* din tabelul *JOB_EMP* și o variabilă de tipul codului job-ului. Să se insereze prin diverse metode 3 înregistrări în tabelul *JOB_EMP*.
14. Creați un tip de date tablou imbricat *DateTab* cu elemente de tip *DATE*. Creați un tabel *FAMOUS_DATES* având o coloană de acest tip. Declarați o variabilă de tip *DateTab* și adăugați-i 5 date calendaristice. Ștergeți al doilea element și apoi introduceți tabloul în tabelul *FAMOUS_DATES*. Selectați-l din tabel. Afișați la fiecare pas.

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

NESTED TABLE *nume_câmp STORE AS* *nume_tabel*;

15. Să se adauge o coloană *info* de tip tablou imbricat în tabelul *DEPT*. Acest tablou are **două componente** în care pentru fiecare departament sunt depuse codul unui angajat și job-ul acestuia. Să se insereze o linie în tabelul imbricat. Să se listeze codurile departamentelor și colecția angajaților corespunzători.
16. Să se creeze un tabel temporar *TEMP_TABLE* cu datele persistente la nivel de sesiune, având o coloană de tip numeric și alta de tip șir de caractere. Prin intermediul unui tablou indexat, să se adauge 500 de linii în acest tabel.
17. Să se insereze o linie nouă în tabelul *EMP*, obținându-se *rowid*-ul acesteia. Să se afișeze valoarea obținută. Măriți cu 30% salariul angajatului cu *rowid*-ul respectiv și obțineți numele și prenumele acestuia. Ștergeți apoi linia corespunzătoare celui *rowid* și afișați informațiile corespunzătoare.
18. Să se declare un tip tablou indexat de numere *T_NUMBERS* și un tip tablou indexat de elemente de tip *T_NUMBERS*, numit *T_MULTINUMBERS*. Să se declare un *TIP T_MULTIVARRAY*

de elemente *T_NUMBERS* și un tip tablou imbricat de elemente *T_NUMBERS*, numit *T_MULTINESTED*. Declarați o variabilă de fiecare din aceste tipuri, inițializați-o și apoi afișați.

19. Definiți un tip tablou imbricat *EmpTab* cu elemente de tipul liniilor tabelului *EMP*. Definiți o variabilă de tip *EmpTab* și apoi inserați linia corespunzătoare în tabelul *EMP*.

20. Declarați un tip *EmpTab* de tip tablou indexat de tablouri imbricate de linii din tabelul *EMPLOYEES*. Declarați o variabilă de acest tip. Inserați într-unul din elementele tabloului informațiile corespunzătoare angajatului având codul 200. Atribuiți valori pentru câmpurile *last_name* și *first_name* ale altui element. Afișați.