Baze de date-Anul 3 (semestrul 1)
Laborator 2 PL/SQL

Tipuri de date compuse

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

I. Înregistrări (RECORD)

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

TYPE *nume_tip* ***IS RECORD***
(*nume_câmp1* {*tip_câmp* | *variabil****ă%TYPE*** |
*nume_tabel.coloan****ă%TYPE*** | *nume_tabel****%ROWTYPE***}
[[***NOT NULL***] {:= | ***DEFAULT***} *expresie1*],
(*nume_câmp2* {*tip_câmp* | *variabil****ă%TYPE*** |
*nume_tabel.coloan****ă%TYPE*** | *nume_tabel****%ROWTYPE***}
[[***NOT NULL***] {:= | ***DEFAULT***} *expresie2*],…);

Ø *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 şi returna sau şterge 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 şteargă angajatul având codul 200 din tabelul *EMP_PNU*. Să se rețină într-o variabilă de tip RECORD codul, numele, salariul şi departamentul acestui angajat (clauza *RETURNING*) . Să se afişeze înregistrarea respectivă. Rollback.

```
DECLARE
  TYPE info_ang_pnu IS RECORD (
    cod_ang  NUMBER(4),
    nume     VARCHAR2(20),
    salariu  NUMBER(8),
    cod_dep  NUMBER(4));
  v_info_ang  info_ang_pnu;
BEGIN
  DELETE FROM emp_pnu
     WHERE employee_id = 200
    RETURNING employee_id, last_name, salary, department_id
      INTO v_info_ang;
  DBMS_OUTPUT.PUT_LINE('A fost stearsa linia continand valorile ' ||
       v_info_ang.cod_ang ||' '||v_info_ang.nume||' ' ||v_info_ang.salariu ||' '
       || v_info_ang.cod_dep) ;
END;
/
ROLLBACK ;
```

2. a) Folosind tipul declarat mai sus, să se adauge o linie în tabelul *EMP_PNU* 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_PNU*.

b) Modificați valoarea unei componente a variabilei definite anterior şi actualizați conținutul liniei introduse în tabel.

```
set serveroutput on
DECLARE
  TYPE info_ang_pnu IS RECORD (
    cod_ang  NUMBER(4):=500,
    nume      VARCHAR2(20):='abc',
    prenume VARCHAR2(20):='john',
    email      emp_pnu.email%TYPE:='abc@mail',
    telefon    emp_pnu.phone_number%type,
    data       emp_pnu.hire_date%TYPE:=SYSDATE,
    job        emp_pnu.job_id%TYPE:='SA_REP',
    salariu    NUMBER(8, 2):=1000,
    comision emp_pnu.commission_pct%TYPE,
    manager  emp_pnu.manager_id%TYPE,
    cod_dep  NUMBER(4):=30
    );
  v_info_ang  info_ang_pnu;
BEGIN
 --inserare; nu ar fi fost posibila maparea unei variabile de tip RECORD într-o lista
-- explicita de coloane
  INSERT INTO emp_pnu
  VALUES    v_info_ang;
  DBMS_OUTPUT.PUT_LINE('A fost introdusa linia continand valorile ' ||
       v_info_ang.cod_ang ||' '||v_info_ang.nume||' ' ||v_info_ang.salariu ||' '
       || v_info_ang.cod_dep) ;
--actualizare
  v_info_ang.nume:='smith';
   UPDATE emp_pnu
   SET ROW=v_info_ang
   WHERE employee_id = v_info_ang.cod_ang;
   DBMS_OUTPUT.PUT_LINE('A fost actualizata linia cu valorile ' ||
       v_info_ang.cod_ang ||' '||v_info_ang.nume||' ' ||v_info_ang.salariu ||' '
       || v_info_ang.cod_dep) ;
END;
/
ROLLBACK ;
```

II. Colecții

Colecțiile permit să fie prelucrate simultan mai multe variabile de acelaşi 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* şi *nested table* pot fi utilizate atât în declarații *PL/SQL*, cât şi în declarații la nivelul schemei (de exemplu, pentru definirea tipului unei coloane a unui tabel relațional).

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

Ø Atribute şi 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	şterge una sau mai multe componente.
EXTEND	Adaugă elemente la sfârşit
LIMIT	Numărul maxim de elemente al unei colecții (pentru vectori), null pentru tablouri imbricate
TRIM	şterge elementele de la sfârşitul unei colecții

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

Ø ***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.*

Ø **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*]*…*

Ø 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 şi comentați exemplul următor. Afişați 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)

Ø 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.

Ø 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 şi 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*.

Ø 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*.

Ø 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*.

Ø Pentru a şterge liniile unui tablou fie se asignează elementelor tabloului valoarea *null*, fie se declară un alt tablou *PL/SQL* (de acelaşi tip) care nu este inițializat şi acest tablou vid se asignează tabloului *PL/SQL* care trebuie şters. În *PL/SQL* 2.3 ştergerea 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 afişeze, apoi să se şteargă tabloul utilizând diverse metode.

```
DECLARE
  TYPE  tablou_numar IS TABLE OF NUMBER
    INDEX BY PLS_INTEGER;
  v_tablou  tablou_numar;
  v_aux     tablou_numar; -- tablou folosit pentru stergere
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 -- metoda 1 de stergere
    v_tablou(i) := NULL;
  END LOOP;
--sau
  v_tablou := v_aux;   -- metoda 2 de stergere
--sau
  v_tablou.delete; --metoda 3 de stergere
  DBMS_OUTPUT.PUT_LINE('tabloul are ' || v_tablou.COUNT ||
                       ' elemente');
END;
/
```

5. Să se definească un tablou de înregistrări având tipul celor din tabelul *dept_pnu*. Să se inițializeze un element al tabloului şi să se introducă în tabelul *dept_pnu*. Să se şteargă elementele tabloului.

```
DECLARE
  TYPE dept_pnu_table_type IS TABLE OF dept_pnu%ROWTYPE
            INDEX BY BINARY_INTEGER;
  dept_table  dept_pnu_table_type;
  i                NUMBER;
BEGIN
  IF dept_table.COUNT <>0 THEN
    i := dept_table.LAST+1;
    ELSE i:=1;
  END IF;
  dept_table(i).department_id := 92;
  dept_table(i).department_name := 'NewDep';
  dept_table(i).location_id := 2700;
  INSERT INTO dept_pnu(department_id, department_name, location_id)
  VALUES (dept_table(i).department_id,
          dept_table(i).department_name,
          dept_table(i).location_id);
  -- sau folosind noua facilitate Oracle9i
  -- INSERT INTO dept_pnu
  -- VALUES  dept_table(i);
  dept_table.DELETE; -- sterge toate elementele
  DBMS_OUTPUT.PUT_LINE('Dupa aplicarea metodei DELETE
     sunt '||TO_CHAR(dept_table.COUNT)||' elemente');
END;
```

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 şi ordinea elementelor este importantă.
- Fiecare element are un index, a cărui limită inferioară este 1.

Ø 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 şi 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 şi utiliza tablourile imbricate şi vectorii, trebuie să declarăm în SQL tipuri de date care să îi reprezinte.

Tablourile imbricate şi 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_pnu* care poate reține maxim 50 de valori de tip VARCHAR2(15).

b) Să se creeze un tabel *test_pnu* având o coloana *cod_ang* de tip NUMBER(4) şi o coloană *proiecte_alocate* de tip *proiect_pnu*. Ce relație se modelează în acest fel?

c) Să se creeze un bloc PL/SQL care declară o variabilă (un vector) de tip *proiect_pnu*, introduce valori în aceasta iar apoi valoarea vectorului respectiv este introdusă pe una din liniile tabelului test_pnu.

La promptul SQL :

```
CREATE TYPE proiect_pnu AS VARRAY(50) OF VARCHAR2(15)
/
CREATE TABLE test_pnu (cod_ang NUMBER(4),
                       proiecte_alocate proiect_pnu);
```

Blocul PL/SQL:

```
DECLARE
        v_proiect proiect_pnu := proiect_pnu(); --initializare utilizând constructorul
BEGIN
        v_proiect.extend (2);
        v_proiect(1) := 'proiect 1';
        v_proiect(2) := 'proiect 2' ;
        INSERT INTO test_pnu VALUES (1, v_proiect);
END;
/
```

8. Să se scrie un bloc care măreşte 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. Se cer 3 soluții.

Soluția 1 :

```
DECLARE
        TYPE t_id IS VARRAY(100) OF emp_pnu.employee_id%TYPE ;
        v_id t_id := t_id();
BEGIN
        FOR contor IN (SELECT * FROM emp_pnu) LOOP
                IF contor. Department_id =50 AND contor.salary < 5000 THEN
                        V_id.extend;
                        V_id(v_id.COUNT) := contor.employee_id;
```

```
            END IF;
        END LOOP;
        FOR contor IN 1..v_id.COUNT LOOP
            UPDATE emp_pnu
            SET salary = salary *1.1
            WHERE employee_id = v_id (contor);
        END LOOP;
END;
/
```

Soluția 2 (varianta FORALL):

```
DECLARE
        TYPE t_id IS VARRAY(100) OF emp_pnu.employee_id%TYPE ;
        v_id t_id := t_id();
BEGIN
        FOR contor IN (SELECT * FROM emp_pnu) LOOP
            IF contor. Department_id =50 AND contor.salary < 5000 THEN
                V_id.extend;
                V_id(v_id.COUNT) := contor.employee_id;
            END IF;
        END LOOP;
        FORALL contor IN 1..v_id.COUNT
            UPDATE emp_pnu
            SET salary = salary *1.1
            WHERE employee_id = v_id (contor);
END;
/
```

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

Soluția 3 (varianta BULK COLLECT):

```
DECLARE
        TYPE t_id IS VARRAY(100) OF emp_pnu.employee_id%TYPE ;
        v_id t_id := t_id();
BEGIN
        SELECT employee_id
        BULK COLLECT INTO v_id
        FROM emp_pnu
        WHERE  department_id =50 AND salary < 5000;
        FORALL contor IN 1..v_id.COUNT
            UPDATE emp_pnu
            SET salary = salary *1.1
            WHERE employee_id = v_id (contor);
END;
/
```

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 ştergere ;
  - 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 acelaşi tip. Valorile de acest tip:
  - pot fi stocate în baza de date,
  - pot fi prelucrate direct în instrucțiuni *SQL*
  - au excepții predefinite proprii.

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

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

Ø Pentru adaugarea de linii intr-un tablou imbricat, acesta trebuie sa fie initializat cu ajutorul **constructorului**.
  - *PL/SQL* apelează un constructor numai în mod explicit.
  - Tabelele indexate nu au constructori.
  - Constructorul primeşte 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 depăşită dimensiunea maximă precizată la declarare.
  - Atunci când constructorul este fără argumente, va crea o colectie fără nici un element (vida), dar care are valoarea *not null*.

Exerciții:

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

```
DECLARE
  TYPE CharTab IS TABLE OF CHAR(1);
  v_Characters CharTab :=
    CharTab('M', 'a', 'd', 'a', 'm', ',', ' ',
          'I', '"', '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. Afişați conținutul acestuia, utilizând metoda EXISTS. Atribuiți valorile tabloului unui tablou index-by. Afişați şi acest tablou, in ordine inversă.

```
DECLARE  -- cod partial, nu sunt declarate tipurile!
  v_NestedTable NumTab := NumTab(-7, 14.3, 3.14159, NULL, 0);
  v_Count BINARY_INTEGER := 1;
  v_IndexByTable IndexByNumTab;
BEGIN
   LOOP
   IF v_NestedTable.EXISTS(v_Count) THEN
```

```
      DBMS_OUTPUT.PUT_LINE(
        'v_NestedTable(' || v_Count || '): ' ||
        v_NestedTable(v_Count));
      v_IndexByTable(v_count) := v_NestedTable(v_count);
      v_Count := v_Count + 1;
    ELSE
      EXIT;
    END IF;
  END LOOP;
-- atribuire invalida
--  v_IndexByTable := v_NestedTable;
  v_Count := v_IndexByTable.COUNT;
  LOOP
    IF v_IndexByTable.EXISTS(v_Count) THEN
      DBMS_OUTPUT.PUT_LINE(
        'v_IndexByTable(' || v_Count || '): ' ||
        v_IndexByTable(v_Count));
      v_Count := v_Count - 1;
    ELSE
      EXIT;
    END IF;
  END LOOP;
END;
END;
/
```

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 şi inițializată anterior.
- *UPDATE* este folosită pentru modificarea unei colecții stocate.
- *DELETE* poate şterge 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 şi operatorul *CAST*. *CAST* are forma sintactică:

  CAST (*nume_colecție* ***AS*** *tip_colecție*)

Exerciții:

13. a) Să se creeze un tip *LIST_ANG_PNU*, de tip vector, cu maxim 10 componente de tip NUMBER(4).

b) Să se creeze un tabel *JOB_EMP_PNU*, având coloanele: cod_job de tip NUMBER(3), titlu_job de tip VARCHAR2(25) şi info de tip *LIST_ANG_PNU*.

c) Să se creeze un bloc PL/SQL care declară şi inițializează două variabile de tip *LIST_ANG_PNU*, o variabilă de tipul coloanei info din tabelul *JOB_EMP_PNU* şi o variabilă de tipul codului job-ului. Să se insereze prin diverse metode 3 înregistrări în tabelul *JOB_EMP_PNU*.

```
CREATE OR REPLACE TYPE list_ang_pnu AS VARRAY(10) OF
                        NUMBER(4)
/
```

```
CREATE TABLE job_emp_pnu (
  cod_job   NUMBER(3),
  titlu_job  VARCHAR2(25),
  info         list_ang_pnu);

DECLARE
  v_list    list_ang_pnu := list_ang_pnu (123, 124, 125);
  v_info_list  list_ang_pnu := list_ang_pnu (700);
  v_info     job_emp_pnu.info%TYPE;
  v_cod      job_emp_pnu.cod_job%TYPE := 7;
  i  INTEGER;
BEGIN
  INSERT INTO job_emp_pnu
  VALUES (5, 'Analist', list_ang_pnu (456, 457));
  INSERT INTO job_emp_pnu
  VALUES (7, 'Programator', v_list);
  INSERT INTO job_emp_pnu
  VALUES (10, 'Inginer', v_info_list);
  SELECT info
  INTO   v_info
  FROM   job_emp_pnu
  WHERE  cod_job = v_cod;
    --afisare v_info
  DBMS_OUTPUT.PUT_LINE('v_info:');
  i := v_info.FIRST;
  while (i <= v_info.last) loop
    DBMS_OUTPUT.PUT_LINE(v_info(i));
    i := v_info.next(i);
  end loop;
END;
/
ROLLBACK;
```

14. Creați un tip de date tablou imbricat *DateTab_pnu* cu elemente de tip *DATE*. Creati un tabel *FAMOUS_DATES_PNU* având o coloană de acest tip. Declarați o variabilă de tip *DateTab_pnu* şi adăugați-i 5 date calendaristice. Ştergeți al doilea element şi apoi introduceți tabloul în tabelul *FAMOUS_DATES_PNU*. Selectați-l din tabel. Afişați 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*;

În SQL*Plus:

```
DROP TABLE famous_dates_pnu;
DROP TYPE DateTab_pnu;
CREATE OR REPLACE TYPE DateTab_pnu AS
  TABLE OF DATE;
/
CREATE TABLE famous_dates_pnu (
  key       VARCHAR2(100) PRIMARY KEY,
  date_list  DateTab_pnu)
  NESTED TABLE date_list STORE AS dates_tab;
```

Blocul PL/SQL:

```
DECLARE
  v_Dates DateTab_pnu := DateTab_pnu(TO_DATE('04-JUL-1776', 'DD-MON-YYYY'),
                        TO_DATE('12-APR-1861', 'DD-MON-YYYY'),
```

```
                    TO_DATE('05-JUN-1968', 'DD-MON-YYYY'),
                    TO_DATE('26-JAN-1986', 'DD-MON-YYYY'),
                    TO_DATE('01-JAN-2001', 'DD-MON-YYYY'));

  -- Procedura locala  pentru afisarea unui  DateTab.
  PROCEDURE Print(p_Dates IN DateTab_pnu) IS
    v_Index BINARY_INTEGER := p_Dates.FIRST;
  BEGIN
    WHILE v_Index <= p_Dates.LAST LOOP
      DBMS_OUTPUT.PUT('  ' || v_Index || ': ');
      DBMS_OUTPUT.PUT_LINE(TO_CHAR(p_Dates(v_Index),
                           'DD-MON-YYYY'));
      v_Index := p_Dates.NEXT(v_Index);
    END LOOP;
  END Print;

BEGIN
  DBMS_OUTPUT.PUT_LINE('Valoarea initiala a tabloului');
  Print(v_Dates);
  INSERT INTO famous_dates_pnu (key, date_list)
    VALUES ('Date importante', v_Dates);
  v_Dates.DELETE(2); -- tabloul va avea numai 4 elemente
  SELECT date_list
    INTO v_Dates
    FROM famous_dates
    WHERE key = 'Date importante';
  DBMS_OUTPUT.PUT_LINE('Tabloul dupa  INSERT si SELECT:');
  Print(v_Dates);
END;
/

ROLLBACK;
```

15. Să se adauge o coloană *info* de tip tablou imbricat în tabelul *DEPT_PNU*. Acest tablou are ***două componente*** în care pentru fiecare departament sunt depuse codul unui angajat şi job-ul acestuia. Să se insereze o linie în tabelul imbricat. Să se listeze codurile departamentelor şi colecția angajaților corespunzători.

```
CREATE OR REPLACE TYPE rec_info_pnu IS OBJECT (cod_ang NUMBER(4), job
VARCHAR2(20));
/
CREATE OR REPLACE TYPE dept_info_pnu IS TABLE OF rec_info_pnu;
/
ALTER TABLE dept_pnu
 ADD (info dept_info_pnu)
 NESTED TABLE info STORE AS info_tab;

SET DESCRIBE DEPTH ALL LINENUM ON INDENT ON

DESC dept_pnu

UPDATE  dept_pnu
SET         info=dept_info_pnu()
WHERE department_id=90;

INSERT INTO TABLE (SELECT info
                FROM   dept_pnu
                WHERE  department_id = 90)
VALUES  (100, 'MyEmp');
```

```
SELECT  d.department_id, t.*
FROM    dept_pnu d, TABLE (d.info) t;
```

16. Să se creeze un tabel temporar *TEMP_TABLE_PNU* cu datele persistente la nivel de sesiune, având o coloană de tip numeric şi alta de tip şir de caractere. Prin intermediul unui tablou indexat, să se adauge 500 de linii în acest tabel. Se cer două variante. Comentați diferențele între variantele propuse.

```
CREATE GLOBAL TEMPORARY TABLE temp_table_pnu (
  num_col NUMBER(6),
  char_col VARCHAR2(20))
ON COMMIT PRESERVE ROWS;
```

```
DECLARE
  TYPE t_Numbers IS TABLE OF temp_table_pnu.num_col%TYPE
    INDEX BY BINARY_INTEGER;
  TYPE t_Chars IS TABLE OF temp_table_pnu.char_col%TYPE
    INDEX BY BINARY_INTEGER;
  v_Numbers t_Numbers;
  v_Chars t_Chars;
BEGIN
  FOR v_Count IN 1..500 LOOP
    v_Numbers(v_Count) := v_Count;
    v_Chars(v_Count) := 'Row number ' || v_Count;
  END LOOP;
  FOR v_Count IN 1..500 LOOP
    INSERT INTO temp_table_pnu VALUES
      (v_Numbers(v_Count), v_Chars(v_Count));
  END LOOP;
END;
/
```

```
DECLARE
  TYPE t_Numbers IS TABLE OF temp_table_pnu.num_col%TYPE
    INDEX BY BINARY_INTEGER;
  TYPE t_Chars IS TABLE OF temp_table_pnu.char_col%TYPE
    INDEX BY BINARY_INTEGER;
  v_Numbers t_Numbers;
  v_Chars t_Chars;
BEGIN
  FOR v_Count IN 1..500 LOOP
    v_Numbers(v_Count) := v_Count;
    v_Chars(v_Count) := 'Row number ' || v_Count;
  END LOOP;
  FORALL v_Count IN 1..500
    INSERT INTO temp_table_pnu VALUES
      (v_Numbers(v_Count), v_Chars(v_Count));
END;
/
```

17. Să se insereze o linie nouă în tabelul *EMP_PNU*, obținându-se *rowid*-ul acesteia. Să se afişeze valoarea obținută. Măriți cu 30% salariul angajatului cu *rowid*-ul respectiv şi obțineți numele şi prenumele acestuia. Ştergeți apoi linia corespunzătoare acelui *rowid* şi afişați informațiile corespunzătoare.

```
DECLARE
```

```
  v_NewRowid ROWID;
  v_FirstName employees.first_name%TYPE;
  v_LastName employees.last_name%TYPE;
  v_ID employees.employee_id%TYPE;
BEGIN
  -- Insereaza o linie noua in tabelul emp_pnu si obtine rowid-ul acesteia
  INSERT INTO emp_pnu
    (employee_id, first_name, last_name, email, job_id, salary, hire_date)
  VALUES
    (emp_sequence.NEXTVAL, 'Xavier', 'Juan','jxav', 'MK_MAN', 2000, SYSDATE)
  RETURNING rowid INTO v_NewRowid;

  DBMS_OUTPUT.PUT_LINE(' rowid-ul noii linii este ' || v_NewRowid);
  UPDATE emp_pnu
    SET salary = salary * 1.3
    WHERE rowid = v_NewRowid
    RETURNING first_name, last_name INTO v_FirstName, v_LastName;
  DBMS_OUTPUT.PUT_LINE('Name: ' || v_FirstName || ' ' || v_LastName);
  DELETE FROM emp_pnu
    WHERE rowid = v_NewRowid
    RETURNING employee_id INTO v_ID;
  DBMS_OUTPUT.PUT_LINE('ID-ul noii linii a fost ' || v_ID);
END;
/
```

18. Să se declare un tip tablou indexat de numere *T_NUMBERS* şi un tip tablou indexat de elemente de tip *T_NUMBERS*, numit *T_MULTINUMBERS*. Să se declare un *TIP T_MULTIVARRAY* de elemente *T_NUMBERS* şi un tip tablou imbricat de elemente *T_NUMBERS*, numit *T_MULTINESTED*. Declarați o variabilă de fiecare din aceste tipuri, inițializați-o şi apoi afişați.

```
DECLARE –acesta este doar codul partial!
  TYPE t_Numbers IS TABLE OF NUMBER
    INDEX BY BINARY_INTEGER;
  TYPE t_MultiNumbers IS TABLE OF t_Numbers
    INDEX BY BINARY_INTEGER;

  TYPE t_MultiVarray IS VARRAY(10) OF t_Numbers;
   TYPE t_MultiNested IS TABLE OF t_Numbers;
  v_MultiNumbers t_MultiNumbers;
BEGIN
  v_MultiNumbers(1)(1) := 12345;
END;
/
```

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

```
DECLARE –cod partial!
  TYPE EmpTab IS TABLE OF emp_pnu%ROWTYPE;
  v_EmpList EmpTab;
BEGIN
  v_EmpList(1) :=
    EmpTab( … );
…
END;
/
```

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* şi *first_name* ale altui element. Afişați.

```
DECLARE
  TYPE EmpTab IS TABLE OF employees%ROWTYPE
    INDEX BY BINARY_INTEGER;
  /* Fiecare element al lui v_Emp este o înregistrare */
  v_Emp EmpTab;
BEGIN
  SELECT *
    INTO v_Emp(200)
    FROM employees
    WHERE employee_id = 200;
  v_Emp(1).first_name := 'Larry';
  v_Emp(1).last_name := 'Lemon';
--afisare …
END;
/
```