

Limbajul PL/SQL- Colecții și înregistrări

Inițializarea și referirea colecțiilor.

-După declarare colecția este cu valoarea NULL.

-Pentru inițializare folosim constructor.

Ex. DECLARE

```
    cursuri_felea lista_cursuri;
```

```
BEGIN
```

```
    cursuri_felea:=lista_cursuri('Baze de date I',  
    'Baze de date II', 'Baze de date orientate obiect');
```

sau

```
cursuri_felea lista_cursuri:=lista_cursuri('Baze de date  
I',...); Putem utiliza NULL.
```

Limbajul PL/SQL- Colecții și înregistrări

Ex. DECLARE

```
proiecte_info lista_proiecte;
```

```
BEGIN
```

```
proiecte_info := lista_proiecte(proiect(1,1000),  
    proiect(2,1500),proiect(3,1600));
```

Constructor fără argumente dă colecție vidă dar NOT NULL.

Ex. DECLARE

```
TYPE clientela IS VARRAY(50) OF Customer;
```

```
clientela_felea clientele:=clientela(); --tablou vid
```

```
BEGIN
```

```
IF clientela_felea IS NOT NULL THEN ... -TRUE
```

Limbajul PL/SQL- Colecții și înregistrări

-Constructorii sunt permisiți în clauze ca
VALUES, SET, componente din SELECT

Ex. BEGIN

```
INSERT INTO depart2 VALUES(  
1, 'info', lista_proiecte(proiect(20, 1000),  
proiect(30, 1400), proiect(40, 12000)));
```

- Referirea colecțiilor

nume_colecție(indice)

indice: 1..2147483647(2**31-1) – tabele nested

1..maximum - varray

Limbajul PL/SQL- Colecții și înregistrări

Ex. DECLARE

```
TYPE filme IS TABLE OF VARCHAR2(30);
filme_rom filme:=filme('Mihai
Viteazul','Rascoala',...);
i BINARY_INTEGER;
x VARCHAR2(30);
BEGIN
i:=1;
IF filme_rom(i)='xxxx' THEN ...
nume_procedura(filme_rom(i));
x:=nume_functie(par1,par2,...,parn)(i)
```

Limbajul PL/SQL- Colecții și înregistrări

- Asignarea colecțiilor: INSERT, UPDATE, FETCH, SELECT, comanda de asignare sau chemare de subprogram.

Ex. DECLARE

```
TYPE clientela1 IS VARRAY(50) OF Customer;
```

```
TYPE clientela2 IS VARRAY(50) OF Customer;
```

```
clientela_soc1 clientela1:=clientela1(...);
```

```
clientela_soc2 clientela1:=clientela1(...);
```

```
clientela_soc3 clientela2:=clientela2(...);
```

```
BEGIN
```

```
clientela_soc2:=clientela_soc1;
```

```
clientela_soc3:=clientela_soc2; -- incorect
```

Limbajul PL/SQL- Colecții și înregistrări

Ex. DECLARE

```
TYPE clientela IS TABLE OF Customer;
```

```
grup1 clientela:=clientela(...);
```

```
grup2 clientela; -- NULL
```

```
grup3 clientela:=clientela(...);
```

BEGIN

```
IF grup1 IS NULL THEN ... --FALSE
```

```
grup1:=grup2;
```

```
IF grup2 IS NULL THEN ... -- TRUE
```

```
grup2:=grup3;
```

```
If grup2 IS NULL THEN ... --FALSE
```

Limbajul PL/SQL- Colecții și înregistrări

- Asignarea elementelor colecțiilor

nume_colectie(indice):=expresie;

Dacă indice este NULL sau neconvertibil la un întreg,
atunci avem excepția VALUE_ERROR

Dacă colecția este NULL, atunci avem excepția
COLLECTION_IS_NULL

Ex. DECLARE

```
TYPE lista IS TABLE OF INTEGER;
```

```
nume1 lista:=lista(100,200,300);
```

```
nume2 lista;
```

Limbajul PL/SQL- Colecții și înregistrări

BEGIN

```
nume1(2):=150;
```

```
nume1(1):=nume1(2);
```

```
nume1(3):=ASCII('G');
```

```
nume1('H'):=400; -- excepție VALUE_ERROR
```

```
nume2(2):= 145; -- excepție COLLECTION_IS_NULL
```

- Colecțiile nu pot fi comparate cu = sau <>
- Colecțiile nu pot apare în DISTINCT, GROUP BY, ORDER

Limbajul PL/SQL- Colecții și înregistrări

```
CREATE TYPE curs AS OBJECT (  
    nr_curs NUMBER(5),  
    titlu   VARCHAR(30),  
    credite NUMBER(1));  
CREATE TYPE list_cursuri AS TABLE OF curs;  
CREATE TABLE depart3 (  
    nume   VARCHAR2(25),  
    director VARCHAR2(30),  
    cursuri list_cursuri)  
NESTED TABLE cursuri STORE AS Tcursuri;
```

Limbajul PL/SQL- Colecții și înregistrări

```
BEGIN
```

```
    INSERT INTO depart3 VALUES
```

```
    ('INFO', 'POPESCU', list_cursuri(curs(1, 'ALGORITMI', 6),  
    curs(2, 'BAZE DE DATE', 6)));
```

```
DECLARE
```

```
noua_lista list_cursuri:=
```

```
    list_cursuri(curs(10, 'Programare', 6),  
                curs(20, 'Logica', 6));
```

```
BEGIN
```

```
    UPDATE depart3
```

```
SET cursuri=noua_lista WHERE nume='INFO';
```

Limbajul PL/SQL- Colecții și înregistrări

Ex.

```
DECLARE
```

```
  cursuri_info list_cursuri;
```

```
BEGIN
```

```
  SELECT cursuri INTO cursuri_info
```

```
    FROM depart3
```

```
    WHERE nume= 'INFO';
```

Limbajul PL/SQL- Colecții și înregistrări

- Prelucrarea elementelor individuale
 - Pentru elementele unei tabele nested – THE
 - Pentru elementele unui VARRAY –proceduri

Ex. BEGIN

```
INSERT INTO
```

```
THE(SELECT cursuri FROM depart3  
WHERE nume='INFO')
```

```
VALUES(22,'Teoria algoritmilor',6);
```

```
UPDATE
```

```
THE(SELECT cursuri FROM depart3  
WHERE nume='INFO')
```

```
SET credite=credite+1
```

```
WHERE nr_curs IN(1,22);
```

Limbajul PL/SQL- Colecții și înregistrări

```
DECLARE
```

```
nr_cursv NUMBER(5);
```

```
titluv VARCHAR2(30);
```

```
BEGIN
```

```
SELECT nr_curs,titlu INTO nr_cursv,titluv
```

```
FROM THE(SELECT cursuri FROM depart3
```

```
WHERE nume='INFO')
```

```
WHERE nr_curs=1;
```

```
DELETE THE(SELECT cursuri FROM depart3
```

```
WHERE nume='INFO')
```

```
WHERE titlu='BD2';
```

Limbajul PL/SQL- Colecții și înregistrări

```
CREATE PROCEDURE adauga_proiect(  
    nr_dept    IN NUMBER;  
    proiect_nou IN Proiect;  
    pozitie    IN NUMBER) AS  
    proiecte_dept lista_proiecte;  
BEGIN  
    SELECT proiecte INTO proiecte_dept FROM depart2  
        WHERE id_dept=nr_dept FOR UPDATE OF proiecte;  
/* Returneaza lista de proiecte in variabila proiecte_dept*/  
    proiecte_dept.EXTENDED;  
/*Extinde tabloul pentru a face loc noului element*/
```

Limbajul PL/SQL- Colecții și înregistrări

```
FOR i IN REVERSE pozitie..proiecte_dept.LAST-1
```

```
LOOP
```

```
    proiecte_dept(i+1):=proiecte_dept(i);
```

```
END LOOP;
```

```
/* Deplaseaza elementele tabloului spre dreapta incepind  
cu indicele pozitie */
```

```
proiecte_dept(pozitie):= proiect_nou;
```

```
/* Insereaza noul proiect */
```

```
UPDATE depart2
```

```
    SET proiecte= proiecte_dept
```

```
    WHERE id_dept= nr_dept;
```

```
/* Actualizeaza tabela depart2 */
```

```
END    adauga_proiect;
```

Limbajul PL/SQL- Colecții și înregistrări

```
CREATE PROCEDURE actualiz_proiect(  
  nr_dept   IN NUMBER;  
  nr_proiect IN NUMBER;  
  cost_nou  IN NUMBER DEFAULT NULL)AS  
  proiecte_dept lista_proiecte;  
BEGIN  
  SELECT proiecte INTO proiecte_dept FROM depart2  
  WHERE id_dept=nr_dept FOR UPDATE OF proiecte;  
  FOR i IN proiecte_dept.FIRST..proiecte_dept.LAST LOOP  
    IF proiecte_dept(i).nr_proiect=nr_proiect THEN
```

Limbajul PL/SQL- Colecții și înregistrări

```
IF cost_nou IS NOT NULL THEN
    proiecte_dept(i).cost:=cost_nou;
END IF;
EXIT;
END IF;
END LOOP;
UPDATE depart2
    SET proiecte=proiecte_dept
    WHERE id_dept=nr_dept;
END actualiz_proiect;
```

Limbajul PL/SQL- Colecții și înregistrări

Ex. Apelul procedurii actualiz_proiect

```
DECLARE
```

```
nr_deptv NUMBER :=10;
```

```
nr_proiectv NUMBER :=100;
```

```
cost_nouv NUMBER :=2000;
```

```
BEGIN
```

```
....
```

```
acualiz_proiect(nr_deptv,nr_proiectv,cost_nouv);
```

```
acualiz_proiect(nr_deptv,nr_proiectv,cost_nou=>1700);
```

Limbajul PL/SQL- Colecții și înregistrări

- Metode ale colecțiilor:
 - EXISTS
 - COUNT
 - LIMIT
 - FIRST, LAST
 - PRIOR, NEXT
 - EXTEND
 - TRIM
 - DELETE

Apelul unei metode:

```
nume_colectie.nume_metoda[(lista_de_parametri)]
```

Limbajul PL/SQL- Colecții și înregistrări

- Numai EXISTS se poate aplica la colecții nule.
- Dacă o altă metodă se aplică la o colecție nulă, atunci se apelează COLLECTION_IS_NULL.

- EXISTS(n) , n este întreg.

Ex. IF cursuri.EXISTS(i) THEN ...

 cursuri(i):=...

- COUNT– returnează nr. de elemente ale colecției.
- Pentru VARRAY COUNT=LAST
- LIMIT –pentru VARRAY dă numărul maxim de elemente, pentru tabele nested are valoarea NULL.
- FIRST, LAST –primul și ultimul indice din colecție.

Pentru VARRAY FIRST =1, LAST=COUNT