**STRUCTURI FUNDAMENTALE DE PROGRAMARE**

**STRUCTURI ALTERNATIVE**

***Structura IF..THEN..END IF***

IF cond1 THEN

secvcom1

ELSE

secvcom2

END IF;

IF cond1 THEN

secvcom1

ELSE

IF cond2 THEN

secvcom2

END IF;

END IF;

* se poate folosi clauza ELSIF în loc de IF imbricate

IF cond1 THEN

secvcom1

ELSIF cond2 THEN secvcom2;

ELSIF cond3 THEN secvcom3;

---------------------------------------

ELSIF condN THEN secvcomN;

ELSE secvcomN+1;

END IF;

**Exemplu:**

În funcție de prețul de listă a produsului având codul citit de la tastatură, se va afişa modificat pe ecran noua valoare.

SET SERVEROUTPUT ON

DECLARE

v\_lista PRODUCT\_INFORMATION.list\_price%type;

BEGIN

SELECT list\_price into v\_lista from PRODUCT\_INFORMATION where product\_id=&p;

dbms\_output.put\_line ('Pretul de lista initial este: '||v\_lista);

IF v\_lista < 500 THEN

v\_lista:=2\* v\_lista;

ELSIF v\_lista between 500 and 1000 THEN

v\_lista:=1.5 \* v\_lista;

ELSE

v\_lista:=1.25\* v\_lista;

END IF;

dbms\_output.put\_line('Pretul final este: '||v\_lista);

end;

/

**Atentie la variabilele de tip NULL si evaluarea in IF!**

De exemplu, în următoarea situaţie se va afişa “Felicitări, sunteţi admis!” din cauza faptului că variabila nota este declarată, dar nu este iniţializată, fiind deci NULL:

DECLARE

nota number;

Begin

IF nota<5 THEN

dbms\_output.put\_line('Ne pare rau, candidatul este respins!');

ELSE

dbms\_output.put\_line('Felicitari, sunteti admis!');

END IF;

END;

/

Observaţi cazurile de mai jos:

X:=10;

Y:=NULL;

IF x!=y then

--intoarce NULL si nu TRUE

END IF;

sau

a:=NULL;

b:=NULL;

IF a=b then

--intoarce NULL si nu TRUE

END IF;

***Structura CASE ... WHEN... THEN...***

Sunt 2 variante:

* expresii CASE (***CASE Expressions***) care intorc un rezultat intr-o variabila. Se termina cu END
* sintaxa CASE (***CASE Statement***) care executa o anumita instructiune. Se termina cu END CASE, iar fiecare rand se termina cu ;

***CASE Expressions:***

Variabila:=

CASE [Selector]

WHEN expression1 THEN result1

WHEN expression2 THEN result2

-----------------------------------------

WHEN expressionN THEN resultN

[ELSE result N+1]

END;

**Exemplu:**

SET SERVEROUTPUT ON

DECLARE

v\_lista PRODUCT\_INFORMATION.list\_price%type;

BEGIN

SELECT list\_price into v\_lista from PRODUCT\_INFORMATION where product\_id=&p;

dbms\_output.put\_line ('Pretul de lista initial este: '||v\_lista);

v\_lista:= CASE WHEN v\_lista < 500 THEN 2\* v\_lista

WHEN v\_lista between 500 and 1000 THEN 1.5 \* v\_lista

ELSE 1.25\* v\_lista END;

dbms\_output.put\_line('Pretul final este: '||v\_lista);

end;

/

***CASE Statement:***

CASE [Selector]

WHEN expression1 THEN action1;

WHEN expression2 THEN action2;

-----------------------------------------

WHEN expressionN THEN actionN;

[ELSE action N+1];

END CASE;

**Exemplu de instrucțiune condițională CASE fără selector:**

SET SERVEROUTPUT ON

DECLARE

v\_lista PRODUCT\_INFORMATION.list\_price%type;

BEGIN

SELECT list\_price into v\_lista from PRODUCT\_INFORMATION where product\_id=&p;

dbms\_output.put\_line ('Pretul de lista initial este: '||v\_lista);

CASE

WHEN v\_lista < 500 THEN

v\_lista:=2\* v\_lista;

WHEN v\_lista between 500 and 1000 THEN

v\_lista:=1.5 \* v\_lista;

ELSE

v\_lista:=1.25\* v\_lista;

END CASE;

dbms\_output.put\_line('Pretul final este: '||v\_lista);

end;

/

**Atenție**: Lipsa clauzei ELSE în cazul CASE STATEMENT poate duce la apariția unei excepții dacă niciuna dintre expresiile din WHEN nu este TRUE. Acest lucru nu se întâmplă în cazul lui IF sau CASE EXPRESSION.

**Structuri repetitive**

***Structura LOOP……END LOOP***

LOOP

Secventa comenzi;

EXIT [WHEN cond];

END LOOP;

**Exemplu**:

Se afişează pe ecran utilizând structura loop…end loop numerele 9,7, 4, 0.

set serveroutput on

DECLARE

v\_nr number(2):=10;

i number(2):=1;

BEGIN

loop

v\_nr:=v\_nr-i;

i:=i+1;

exit when v\_nr < 0;

dbms\_output.put\_line(v\_nr);

end loop;

END;

/

Se afişează în ordine angajații cu codurile în intervalul 100-110 atât timp cât salariul acestora este mai mic decât media:

DECLARE  
v\_sal employees.salary%type;  
v\_salMediu v\_sal%type;  
i number(4):=100;  
BEGIN  
SELECT avg(salary) into v\_salmediu from employees;  
dbms\_output.put\_line('Salariul mediu este: '||v\_salmediu);  
loop  
select salary into v\_sal from employees where employee\_id=i;  
dbms\_output.put\_line('Salariatul cu codul '||i||' are salary: '||v\_sal);  
i:=i+1;  
exit when v\_sal<v\_salmediu or i>110;  
end loop;  
end;  
/

***Structura WHILE…..LOOP….END LOOP***

WHILE cond LOOP

Secventa comenzi 1;

Secventa comenzi 2;

EXIT [WHEN cond];

END LOOP;

**Exemplu**:

Se afişează pe ecran utilizând structura while loop…end loop numerele 9,7, 4, 0.

set serveroutput on

DECLARE

v\_nr number(2):=10;

i number(2):=1;

BEGIN

while v\_nr > 0

loop

v\_nr:=v\_nr-i;

i:=i+1;

dbms\_output.put\_line(v\_nr);

end loop;

END;

/

Se afişează în ordine angajatii cu codurile în intervalul 100-110 atât timp cât salariul acestora este mai mic decât media:

DECLARE  
v\_sal employees.salary%type;  
v\_salMediu v\_sal%type;  
i number(4):=100;  
BEGIN  
SELECT avg(salary) into v\_salmediu from employees;  
dbms\_output.put\_line('Salariul mediu este: '||v\_salmediu);  
while i<=110 loop  
select salary into v\_sal from employees where employee\_id=i;  
dbms\_output.put\_line('Salariatul cu codul '||i||' are salary: '||v\_sal);  
i:=i+1;  
exit when v\_sal<v\_salmediu;  
end loop;  
end;  
/

***Structura FOR…..LOOP….END LOOP***

FOR var IN [REVERSE] valmin..valmax LOOP

Secventa comenzi;

EXIT [WHEN cond];

END LOOP;

**Observatii**:

Valorile intervalului pot fi de orice tip, dar sa aiba valori care pot fi convertite la un intreg (de exemplu 20/13 sau 11/5). Daca aceste 2 valori vor fi egale ca intregi atunci instructiunile din interiorul ciclului se executa o singura data. De exemplu secventa:

FOR i IN 3..3 LOOP

Secventa comenzi;

END LOOP;

**Exemple:**

Se afişează pe ecran utilizând structura for…end loop numerele 9,7, 4, 0.

set serveroutput on

DECLARE

v\_nr number(2):=10;

i number(2);

BEGIN

for i in 1..10

loop

v\_nr:=v\_nr-i;

exit when v\_nr < 0;

dbms\_output.put\_line(v\_nr);

end loop;

END;

/

Se afişează în ordine angajatii cu codurile în intervalul 100-110 atât timp cât salariul acestora este mai mic decât media:

DECLARE  
v\_sal employees.salary%type;  
v\_salMediu v\_sal%type;  
-- i nu mai trebuie declarat  
BEGIN  
SELECT avg(salary) into v\_salmediu from employees;  
dbms\_output.put\_line('Salariul mediu este: '||v\_salmediu);  
for i in 100..110 loop  
select salary into v\_sal from employees where employee\_id=i;  
dbms\_output.put\_line('Salariatul cu codul '||i||' are salariul: '||v\_sal);  
exit when v\_sal<v\_salmediu;  
end loop;  
end;  
/

Să se afişeze numărul de comenzi ale fiecărui angajat al cărui id este situat în intervalul 155..160, dar să se întrerupă afişarea în cazul în care se găseşte primul angajat din acest interval care nu are nici o comandă:

DECLARE

v\_nr number;

v\_nume employees.last\_name%type;

v\_id employees.employee\_id%type;

BEGIN

for v\_id in 155..160 loop

v\_nr:=0;

SELECT count(c.order\_id) into v\_nr from orders c,employees a

where c.sales\_rep\_id=a.employee\_id and a.employee\_id=v\_id;

dbms\_output.put\_line('Salariatul cu id-ul: '||v\_id||' are: '||v\_nr||' comenzi');

exit when v\_nr=0;

end loop;

end;

/

**Să se încarce în tabela MESAJE numere de la 1…10 cu excepţia lui 6 şi 8.**

CREATE TABLE MESAJE (REZ VARCHAR2(30));

BEGIN

FOR i IN 1..10 LOOP

IF i=6 OR i=8 THEN

NULL;

ELSE

INSERT INTO MESAJE(REZ) VALUES (i);

END IF;

COMMIT;

END LOOP;

END;

/

***Structuri LOOP imbricate (se vor eticheta loop-urile)***

BEGIN

<<LOOP\_EXTERN>>

LOOP

v\_var:=v\_var+1;

EXIT WHEN v\_var>10;

<<LOOP\_INTERN>>

LOOP

……………………….

EXIT LOOP\_EXTERN WHEN cond1;

EXIT WHEN cond2;

…………………………

END LOOP;

…………………………….

END LOOP;

END;

**TIPURI DE DATE COMPUSE**

**1. Tipul înregistrare (RECORD)**

* Reprezintă un grup de date logic corelate (de exemplu, datele despre un client: codc, nume, adresa sunt diferite ca tip dar corelate logic). Ele pot fi manipulate ca o unitate;
* Fiecare câmp (element al structurii) are un nume şi un tip de dată;
* Dacă declarăm o variabilă de un tip înregistrare, câmpurile sunt referite **NUME\_VARIABILĂ.NUME\_CÂMP.**

**TYPE NUME\_TIP IS RECORD (nume\_câmp TIP\_DATA [,nume\_câmp TIP\_DATA:=|DEFAULT valoare]...);**

**NUME\_VARIABILĂ NUME\_TIP;**

**Exemplu**:

Utilizând un tip de dată înregistrare definit de utilizator, să se afișeze preţul minim al produsului cu codul 3133.

DECLARE

TYPE tip\_produse IS RECORD

(v\_cod PRODUCT\_INFORMATION.product\_id%type NOT NULL:=3000,

v\_den PRODUCT\_INFORMATION.product\_name%type,

v\_pret\_min PRODUCT\_INFORMATION.min\_price%type);

VREC\_PROD TIP\_PRODUSE;

BEGIN

SELECT product\_id, product\_name, min\_price into vrec\_prod from PRODUCT\_INFORMATION where product\_id=3133;

dbms\_output.put\_line('Produsul: '|| vrec\_prod.v\_den|| ' are pretul minim: '||vrec\_prod.v\_pret\_min);

END;

/

Pentru a defini o variabilă tip înregistrare pe baza coloanelor unei tabele se folosește **%ROWTYPE**. In acest caz numele elementelor au același nume ca şi coloanele tabelei, acelaşi tip de date şi se găsesc în aceeaşi ordine.

**NUME\_VARIABILA TABELA%ROWTYPE;**

**Exemple:**

Utilizând un tip de dată înregistrare de același tip cu tabela produse să se afişeze preţul minim al produsului cu codul 3133.

DECLARE

vrec\_prod PRODUCT\_INFORMATION%rowtype;

BEGIN

SELECT \* into vrec\_prod from PRODUCT\_INFORMATION where product\_id=3133;

dbms\_output.put\_line('Produsul: '|| vrec\_prod.product\_name|| ' are pretul minim: '||vrec\_prod.min\_price);

END;

/

Utilizând un tip de dată înregistrare de același tip cu un rând din tabela DEPARTAMENTE să se afișeze denumirea fiecărui departament cu id-ul: 10, 20, 30, 40, 50.

DECLARE

vrec\_dep departments%rowtype;

i number:=10;

BEGIN

loop

SELECT \* into vrec\_dep from departments where department\_id=i;

dbms\_output.put\_line('Departamentul: '|| vrec\_dep.department\_id|| ' are denumirea de: '||vrec\_dep.department\_name);

exit when i>=50;

i:=i+10;

end loop;

END;

/

**2. Tipuri de date colecție**

* **Tablouri indexate** (index-by tables/associative arrays) – sunt similare tabelelor de dispersie din alte limbaje de programare;
* **Tablouri** **imbricate** (nested-tables) – similare cu mulțimile din alte limbaje de programare;
* **Vectori de lungime variabilă (**VARRAYS) – similari cu vectorii din alte limbaje de programare.

Tablourile indexate se pot folosi doar în blocuri PL/SQL. Tablourile imbricate precum și vectorii de lungime variabilă pot fi folosiți și la nivel de schemă - se pot defini folosind CREATE TYPE tipuri SQL și mai apoi coloane în tabele de respectivul tip.

**Sintaxă declarare tablouri indexate:**

**TYPE NUME\_TAB IS TABLE OF {TIP\_DATA [variabila%type | tabela.coloana%type [NOT NULL]| tabela%rowtype}**

**INDEX BY PLS\_INTEGER|BINARY\_INTEGER|VARCHAR2(dimensiune);**

**V\_TAB NUME\_TAB;**

* Adresarea se realizează cu **v\_tab(index).câmp**;
* TIP\_DATA poate fi orice tip PL/SQL cu excepția REF CURSOR;
* Indexul este unic, dar în ordine aleatorie şi poate fi negativ. Intervalul pentru PLS\_INTEGER este (-2147483647, 2147483647);
* Se pot utiliza următoarele proprietăţi şi metode:

v\_tab**.**EXISTS(i)

v\_tab**.**COUNT

v\_tab**.** FIRST şi v\_tab**.**LAST

v\_tab**.** PRIOR(i) şi v\_tab.NEXT(i)

v\_tab**.**DELETE sau v\_tab**.**DELETE(i) sau v\_tab**.**DELETE(i,j)

**Exemplu**:

*--Utilizarea unui tablou indexat de tipul PRODUSE.DENPRODUS*

DECLARE

--declarare tip

type num\_table is table of PRODUCT\_INFORMATION.product\_name%type index by pls\_integer;

-- declarare variabilă tablou

v\_tab num\_table;

i number(5):=2252;

BEGIN

--incarcarea in tablou:

loop

SELECT product\_name into v\_tab(i) from PRODUCT\_INFORMATION where product\_id=i;

i:=i+1;

exit when i>2255;

end loop;

--extragerea din tablou

for i in v\_tab.first..v\_tab.last loop

IF v\_tab.EXISTS(i) then

dbms\_output.put\_line('Nume produs: '|| v\_tab(i));

end if;

end loop;

dbms\_output.put\_line('Total produse in tabloul indexat: '|| v\_tab.count);

END;

/

**Exemplu**:

*--Utilizarea unui tablou indexat de același tip cu un rând din tabela EMPLOYEES - %ROWTYPE*

DECLARE

--declararea tipului si a variabilei

type ang\_table is table of employees%rowtype index by pls\_integer;

v\_tab ang\_table;

BEGIN

--incarcarea in tablou:

for i in 130..135 loop

SELECT \* into v\_tab(i) from employees where employee\_id=i;

end loop;

--extragerea din tablou

for i in v\_tab.first..v\_tab.last loop

dbms\_output.put\_line('Angajatul: '|| v\_tab(i).last\_name|| ' lucreaza in departamentul: '||v\_tab(i).department\_id);

end loop;

dbms\_output.put\_line('Total angajati in tabloul indexat: '|| v\_tab.count);

END;

/

**EXERCITIU**

Fie un șir de cifre stocat într-o variabilă de tip VARCHAR2. Să se parcurgă șirul iar fiecare cifră să se stocheze într-o colecție de tip index-by-table. Să se șteargă al treilea element din colecție. Să se parcurgă colecția cu FOR și cu WHILE – se va include **cod + captură de eran**

**Raspuns:**

**SET SERVEROUTPUT ON**

**DECLARE**

**V\_STRING VARCHAR2(100):='25478624578965213475';**

**TYPE T\_IBT IS TABLE OF VARCHAR2(1) INDEX BY PLS\_INTEGER;**

**V T\_IBT;**

**I PLS\_INTEGER;**

**BEGIN**

**FOR I IN 1..LENGTH(V\_STRING) LOOP**

**V(I):=SUBSTR(V\_STRING,I,1);**

**END LOOP;**

**DBMS\_OUTPUT.PUT\_LINE('The no. of elements is:'||V.COUNT);**

**FOR I IN V.FIRST..V.LAST LOOP**

**DBMS\_OUTPUT.PUT\_LINE('The element with the index '||I||' has the value:'||V(I));**

**END LOOP;**

**V.DELETE(3);**

**DBMS\_OUTPUT.PUT\_LINE('The no. of elements is:'||V.COUNT);**

**DBMS\_OUTPUT.PUT\_LINE('FOR loop:');**

**FOR I IN V.FIRST..V.LAST LOOP**

**IF V.EXISTS(I) THEN**

**DBMS\_OUTPUT.PUT\_LINE('The element with the index '||I||' has the value:'||V(I));**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE('The element with the index '||I||' has been deleted!');**

**END IF;**

**END LOOP;**

**DBMS\_OUTPUT.PUT\_LINE('WHILE loop:');**

**I:=V.FIRST;**

**WHILE I IS NOT NULL LOOP**

**IF V.EXISTS(I) THEN**

**DBMS\_OUTPUT.PUT\_LINE('The element with the index: '||I||' has the value: '||V(I));**

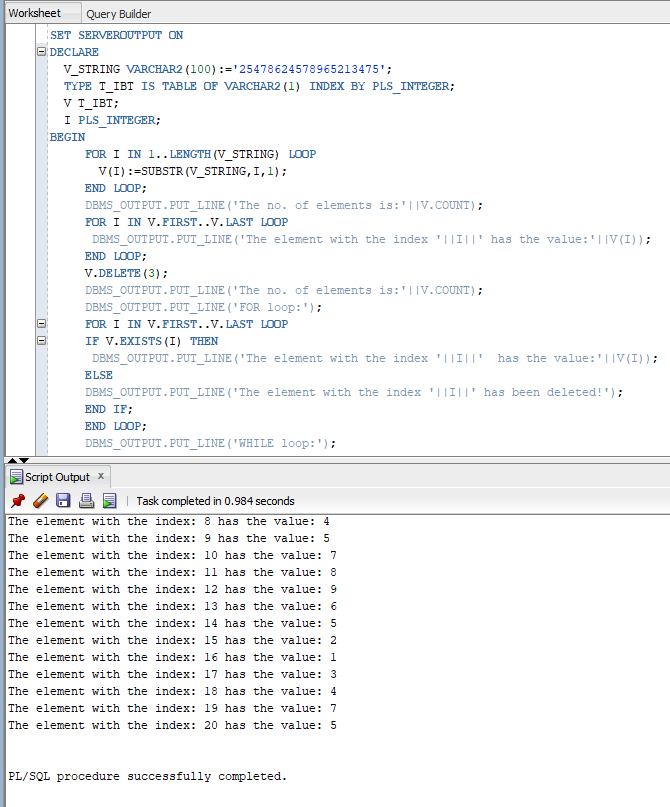
**END IF;**

**I:=V.NEXT(I);**

**END LOOP;**

**END;**

**/**



**Tablouri** **imbricate** (nested-tables)

Intr-un tablou imbricat, nu putem construi elemente doar specificând indexul.

SET SERVEROUTPUT ON

DECLARE

*--declarare tip*

type t\_nt is table of number;

*-- declarare variabilă tablou (colectie)*

v t\_nt;

BEGIN

v(1):=10;

dbms\_output.put\_line('Numar de elemente: '||v.count);

END;

/

ORA-06531: Reference to uninitialized collection

ORA-06512: at line 7

Trebuie mai întâi să extindem colecția:

SET SERVEROUTPUT ON

DECLARE

*--declarare tip*

type t\_nt is table of number;

*-- declarare variabilă tablou (colectie)*

v t\_nt;

BEGIN

v:=t\_nt();

v.extend(1);

v(1):=10;

dbms\_output.put\_line('Numar de elemente: '||v.count);

v.extend(7);

dbms\_output.put\_line('Numar de elemente: '||v.count);

v(8):=120;

for i in v.first..v.last loop

dbms\_output.put\_line(i||'->'||v(i));

end loop;

END;

/

Elementele de la 2 la 7 există, dar nu au valori.

Numar de elemente: 1

Numar de elemente: 8

1->10

2->

3->

4->

5->

6->

7->

8->120

PL/SQL procedure successfully completed

SET SERVEROUTPUT ON

DECLARE

*--declarare tip*

type t\_nt is table of number;

*-- declarare variabilă tablou (colectie)*

v t\_nt;

BEGIN

v:=t\_nt();

v.extend(1);

v(1):=10;

dbms\_output.put\_line('Numar de elemente: '||v.count);

v.extend(7);

dbms\_output.put\_line('Numar de elemente: '||v.count);

v(8):=120;

v.delete(2,7); *-- se sterg elementele cu indexul intre 2 si 7*

for i in v.first..v.last loop

dbms\_output.put\_line(i||'->'||v(i));

end loop;

END;

/

Rularea va returna o excepție:

ORA-01403: no data found

ORA-06512: at line 16

**EXERCITIU**

Rezolvați problema astfel încât elementele care nu există să nu fie afișate. Se vor folosi 2 variante:

1. Cu WHILE și NEXT
2. Cu FOR și EXISTS

Se va include **cod + captură de eran** pentru ambele variante

Raspuns:

**SET SERVEROUTPUT ON**

**DECLARE**

**type t\_nt is table of number;**

**v t\_nt;**

**I NUMBER;**

**BEGIN**

**v:=t\_nt();**

**v.extend(1);**

**v(1):=10;**

**dbms\_output.put\_line('No of elements: '||v.count);**

**v.extend(7);**

**dbms\_output.put\_line('No of elements: '||v.count);**

**v(8):=120;**

**v.delete(2,7);**

**I:=V.FIRST;**

**WHILE I IS NOT NULL LOOP**

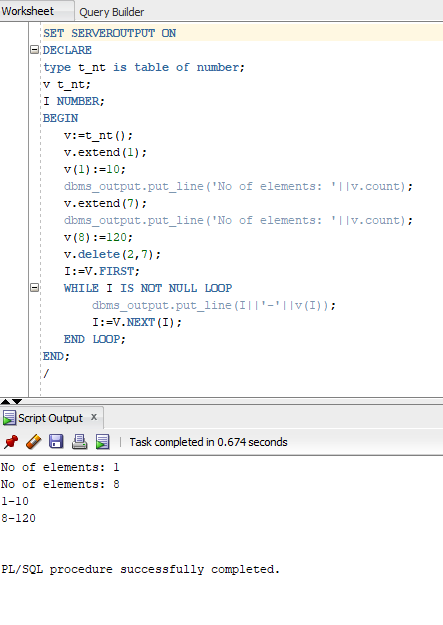
**dbms\_output.put\_line(I||'-'||v(I));**

**I:=V.NEXT(I);**

**END LOOP;**

**END;**

**/**



**SET SERVEROUTPUT ON**

**DECLARE**

**type t\_nt is table of number;**

**v t\_nt;**

**I NUMBER;**

**BEGIN**

**v:=t\_nt();**

**v.extend(1);**

**v(1):=10;**

**dbms\_output.put\_line('No of elements: '||v.count);**

**v.extend(7);**

**dbms\_output.put\_line('No of elements: '||v.count);**

**v(8):=120;**

**v.delete(2,7);**

**for i in v.first..v.last loop**

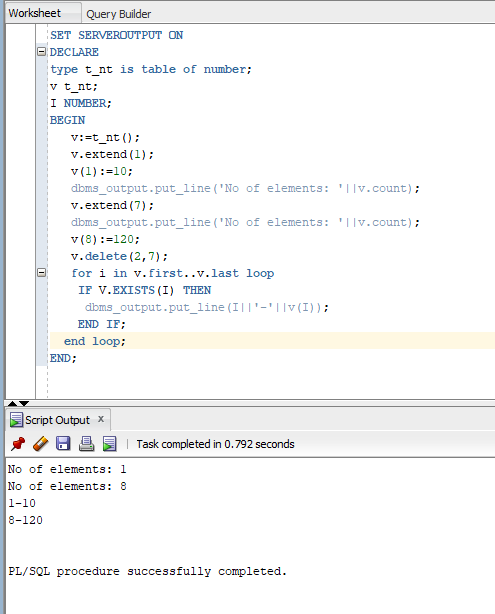
**IF V.EXISTS(I) THEN**

**dbms\_output.put\_line(I||'-'||v(I));**

**END IF;**

**end loop;**

**END;**



**Vectori de lungime variabilă (**VARRAYS)

SET SERVEROUTPUT ON

DECLARE

TYPE tab\_vec IS VARRAY(10) OF NUMBER; /\* vectorul nu va putea avea mai mult de 10 elemente\*/

t tab\_vec := tab\_vec();

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Colectia are ' || t.COUNT ||' elemente');

FOR i IN 1..10 LOOP

t.EXTEND;

t(i):=i;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Colectia are ' || t.COUNT ||' elemente');

FOR i IN t.FIRST..t.LAST LOOP

DBMS\_OUTPUT.PUT(t(i) || ' ');

END LOOP;

DBMS\_OUTPUT.NEW\_LINE;

FOR i IN 1..10 LOOP

IF i mod 2 = 1 THEN t(i):=null;

END IF;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Colectia are ' || t.COUNT ||' elemente');

FOR i IN t.FIRST..t.LAST LOOP

DBMS\_OUTPUT.PUT(nvl(t(i), 0) || ' ');

END LOOP;

DBMS\_OUTPUT.NEW\_LINE;

t := tab\_vec(10,20,30); *-- se va suprascrie*

DBMS\_OUTPUT.PUT\_LINE('Colectia are ' || t.COUNT ||' elemente');

t.DELETE;

DBMS\_OUTPUT.PUT\_LINE('Colectia are ' || t.COUNT ||' elemente');

t:=null;

*--DBMS\_OUTPUT.PUT\_LINE('Colectia are ' || t.COUNT ||' elemente'); -->Reference to uninitialized collection*

END;

/

SET SERVEROUTPUT ON

DECLARE

TYPE tab\_vec IS VARRAY(10) OF NUMBER;

t tab\_vec := tab\_vec();

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Colectia are ' || t.COUNT ||' elemente');

FOR i IN 1..11 LOOP

t.EXTEND;

t(i):=i;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Colectia are ' || t.COUNT ||' elemente');

END;

/

ORA-06532: Subscript outside of limit

ORA-06512: at line 7

Excepția apare deoarece nu putem avea mai mult de 10 elemente in vector (valoarea din TYPE).

Toate tipurile de colecții pot fi folosite în clauza BULK COLLECT a SELECT sau RETURNING a unei instrucțiuni INSERT, UPDATE sau DELETE.

La VARRAY trebuie sa fim atenți să nu fie mai multe elemente decât dimensiunea maximă:

SET SERVEROUTPUT ON

DECLARE

TYPE tab\_vec IS VARRAY(5) OF countries.country\_name%type;

v tab\_vec;

BEGIN

SELECT country\_name BULK COLLECT INTO v FROM countries WHERE region\_id=1;

END;

/

ORA-22165: given index [6] must be in the range of [1] to [5]

ORA-06512: at line 5

DECLARE

TYPE tab\_vec IS VARRAY(10) OF countries.country\_name%type;

v tab\_vec;

BEGIN

SELECT country\_name BULK COLLECT INTO v FROM countries WHERE REGION\_id=1;

for i in 1..v.count loop

dbms\_output.put\_line(v(i));

end loop;

END;

/

De aceea, se recomandă a se folosi cu BULK COLLECT sau RETURNING colecții de tip index-by-table sau nested-table.

SET SERVEROUTPUT ON

DECLARE

TYPE tab\_tari IS VARRAY(10) OF countries.country\_name%type;

TYPE tab\_produse IS table of product\_information%rowtype;

v tab\_tari;

p tab\_produse;

BEGIN

SELECT country\_name BULK COLLECT INTO v FROM countries WHERE REGION\_ID=1;

for i in 1..v.count loop

dbms\_output.put\_line(v(i));

end loop;

delete from PRODUCT\_INFORMATION where product\_id not in (select product\_id from order\_items)

returning product\_id,product\_name, product\_description, category\_id, weight\_class, supplier\_id,

product\_status, list\_price, min\_price, catalog\_url

BULK COLLECT INTO p;

dbms\_output.put\_line('S-au sters '||p.count||' produse');

/\* Sa se afiseze denumirea si categoria produselor care au fost sterse\*/

rollback;

END;

/

Se va include **cod + captură de eran**

**SET SERVEROUTPUT ON**

**DECLARE**

**TYPE countries\_table IS table of countries.country\_name%type;**

**TYPE products\_table IS table of product\_information%rowtype;**

**v countries\_table;**

**p products\_table;**

**BEGIN**

**SELECT country\_name BULK COLLECT INTO v FROM COUNTRIES WHERE REGION\_ID = 1;**

**for i in 1..v.count loop**

**dbms\_output.put\_line(v(i));**

**end loop;**

**delete from PRODUCT\_INFORMATION where product\_id not in (select product\_id from order\_items)**

**returning product\_id,product\_name, product\_description, category\_id, weight\_class, supplier\_id,**

**product\_status, list\_price, min\_price, catalog\_url**

**BULK COLLECT INTO p;**

**dbms\_output.put\_line('S-au sters '||p.count||' produse');**

**FOR I IN P.FIRST..P.LAST LOOP**

**DBMS\_OUTPUT.PUT\_LINE('Product'||I||' with name '||P(I).PRODUCT\_NAME||' FROM CATEGORY '||P(I).CATEGORY\_ID || 'has been deleted!');**

**END LOOP;**

**rollback;**

**END;**

**/**

