**NULL**

Daca in if ne da null => nu se mai executa secventa de comenzi

Null combinat cu altcv rezulta null, iar null negat e tot null

ex: if null <> 5 then ... else “aici intra”

if null = null then .. else “aici intra”

**Tipul inregistrare: RECORD**

= grup de date stocate sub forma de campuri (mai multe coloane cu mai multe randuri de inregistrari)

* se poate aplica pe el insert, update cu set rows, returning pt a anula modificarile LMD facute

TYPE emp\_record IS RECORD (cod employees.employee\_id%TYPE,

salariu employees.salary%TYPE,

job employees.job\_id%TYPE);

v\_ang emp\_record;

Apoi merge sa salvezi in el atatea coloane cate i-ai definit:

SELECT employee\_id, salary, job\_id

INTO v\_ang – fara bulk collect into

* se poate defini cu tipul unui alt tabel existent => se copiaza structura (campurile) si tipurile de date din ea. ex: variabila cu aceeasi structura ca tabelulu employees

v\_ang1 employees%ROWTYPE;

**Tablouri indexate**

-> nu au constructor

-- tablou indexat de numere.

TYPE tablou\_indexat IS TABLE OF NUMBER INDEX BY PLS\_INTEGER;

t tablou\_indexat;

-- un tablou indexat de înregistrări având tipul celor din tabelul emp\_\*\*\*

TYPE tablou\_indexat IS TABLE OF emp\_\*\*\*%ROWTYPE INDEX BY BINARY\_INTEGER;

t tablou\_indexat;

**Tablouri imbricate**

= multime neordonata de elem de acelasi tip! trb initializate cu un constructor

* diferenta dintre ele e ca aici nu exista clauza index by
* la inceput e dens, dar dupa stergere pot ramane spatii goale. Se fol extend pt a adauga un el nou si next pt a merge la elem urmator

TYPE tablou\_imbricat IS TABLE OF NUMBER;

t tablou\_imbricat := tablou\_imbricat();

--tablou imbricat de caractere și o variabilă de acest tip.

declare

TYPE tablou\_imbricat IS TABLE OF CHAR(1);

t tablou\_imbricat := tablou\_imbricat('m', 'i', 'n', 'i', 'm');

begin

i := t.FIRST;

WHILE i <= t.LAST LOOP

DBMS\_OUTPUT.PUT(t(i));

i := t.NEXT(i);

END LOOP;

end;

**Vectori**

= la rel one-to-many

* spre deosebire de tablourile imbricate, vectorii au o dimensiune maximă stabilită la declarare.

TYPE vector IS VARRAY(20) OF NUMBER;

t vector:= vector();

* Obs. Comanda FORALL permite ca toate liniile unei colecții să fie transferate simultan printr-o

singură operație. Procedeul este numit bulk bind.

FORALL i IN coduri.FIRST..coduri.LAST

DELETE FROM emp\_\*\*\*

WHERE employee\_id = coduri (i);

**Operatii pe vectori/tablouri**

* obs: nu se pot aplica pe null (in afara de exists)

.count

.delete()

.exists(n) -> true daca exista al n-lea elem din colectia respectiva POATE FI APLICAT PE NULL!

.first, .last

.next(n), .prior(n) -> indicele elem urmator, respectiv precent

.extend -> adauga un element null la sfarsit => .extend(n) adauga n elem null; extend(n,i) adauga n copii ale el i

.limit -> nr maxim de elem

.trim -> sterge de la sfarsitul colectiei; .trim(n) sterge ultimele n elem

**Prelucrarea colecțiilor stocate în tabele**

* Vector stocat într-un tabel
* e prelucrat ca un intreg (nu pot fi modif elem individuale, decat daca se selecteaza vectorul intr-o variabila PL/SQL, se modifica valoarea sa si se insereaza inapoi in tabel)
* Tablou imbricat stocat într-un tabel
* poate fi prelucrat si ca intreg, dar si la nivel de elem individuale, folosind functia *TABLE*

ex:

-- selectie elemente colectie

SELECT \* FROM TABLE (SELECT grupe FROM raion\_grupe\_imb WHERE id\_categorie = 1);

--adaugare element in colectie

INSERT INTO TABLE (SELECT grupe FROM raion\_grupe\_imb WHERE id\_categorie = 1) VALUES ('r13'); -- sau aici merge tot o subcerere ex:

-- adaugare elemente obtinute cu subcerere

INSERT INTO TABLE (SELECT grupe FROM raion\_grupe\_imb WHERE id\_categorie = 1)

SELECT denumire

FROM categorii

WHERE id\_parinte = 1;

**Procedeul bulk collect**

= permite transferul liniilor intre SQL si PL/SQL prin intermediul colectiilor (motorul PL/SQL comunică motorului SQL să obțină mai multe linii odată și să le plaseze într-o colecție; motorul SQL regăsește toate liniile și le încarcă în colecție, după care predă controlul motorului PL/SQL)

* ex:

DECLARE

TYPE t\_ind IS TABLE OF categorii.id\_categorie%TYPE INDEX BY PLS\_INTEGER;

TYPE t\_imb IS TABLE OF categorii.denumire%TYPE;

TYPE t\_vec IS VARRAY(10) OF categorii.nivel%TYPE;

v\_ind t\_ind;

v\_imb t\_imb;

v\_vec t\_vec;

BEGIN

SELECT id\_categorie, denumire, nivel BULK COLLECT INTO v\_ind, v\_imb, v\_vec

FROM categorii WHERE ROWNUM <= 10;

FOR i IN 1..10 LOOP

DBMS\_OUTPUT.PUT(v\_ind(i) || ' ' || v\_imb(i) || ' ' ||v\_vec(i)); DBMS\_OUTPUT.NEW\_LINE;

END LOOP;

END;

**Procedeul bulk bind**

= permite transferul liniilor din colecție printr-o singură operație, cu ajutorul comenzii FORALL (atentie la restrictii!)