

# BAZE DE DATE ORIENTATE OBIECT

# LIMBAJUL PL/SQL

- Suport pentru SQL
- Suport pentru programarea orientata obiect
- Performanta buna
- Portabilitate
- Productivitate mare
- Integrarea cu Oracle

# Limbajul PL/SQL- Tabele

- CREATE TABLE dept(deptno NUMBER(2) NOT NULL, dname VARCHAR2(14), loc VARCHAR2(13))
- CREATE TABLE emp(empno NUMBER(4) NOT NULL, ename VARCHAR2(10), job VARCHAR2(9), mgr NUMBER(4), hiredate DATE, sal NUMBER(7,2), comm NUMBER(7,2), deptno NUMBER(2))

# Limbajul PL/SQL- Elemente de bază

- Caractere de bază:

A..Z, a..z

0..9

tab, space, CR

Simboli ☹) + - \* / < > = ! ~ ; : . ' @ % , " # \$ & \_ { } ? [ ]

Nu este case senzitive

# Limbajul PL/SQL- Elemente de bază

- Unitati lexicale:
  - delimitatori
  - identificatori
  - literali
  - comentarii

Unitatile lexicale sunt separate prin spatii

# Limbajul PL/SQL- Elemente de bază

- Delimitatori:

simboli simpli: +, \*, %, etc.

simboli compusi: \*\*, <>, !=, ~=, <=, >=, :=, =>

.., ||, <<, >>, --, ?\*, \*/

Identificatori:

Formati din litere, numerale, \$, \_, semne numerice

# Limbaajul PL/SQL- Elemente de bază

- Cuvinte rezervate:  
DECLARE, BEGIN , END, BOOLEAN, ..
- Identificatori predefiniti:  
INVALID\_NUMBER,...
- Identificatori cu “  
“sir caractere”

Ex. TYPE cuvint rezervat,

SELECT numar, type, ....-eroare

SELECT numar, "TYPE",...

# Limbajul PL/SQL- Elemente de bază

- Literali numerici:
  - sir de cifre, eventual precedat de +, sau –
  - notatia stiintifica cu E sau e
  - operatorul \*\*
- Literali de tip caracter:
  - ‘c’
- Literali de tip sir:
  - ‘sir de caractere’

Literalii de tip sir, cu exceptia sirului nul au tipul de date CHAR.

# Limbajul PL/SQL- Elemente de bază

- Literali booleeni:  
TRUE, FALSE, NULL
- Comentarii:  
--  
\*/comentarii  
....  
/\*

# Limbajul PL/SQL- Elemente de bază

- Tipuri de date:

- Scalare:

BINARY\_INTEGER

DEC

DECIMAL

DOUBLE PRECISION

FLOAT

INT

INTEGER

NATURAL

NATURALN

NUMBER

NUMERIC

PLS\_INTEGER

POSITIVE

POSITIVEN

REAL

SIGNTYPE

# Limbajul PL/SQL- Elemente de bază

Tipuri scalare:

CHAR

ROWID

CHARACTER

STRING

LONG

VARCHAR

LONG RAW

VARCHAR2

NCHAR

BOOLEAN

NVARCHAR2

DATE

RAW

# Limbajul PL/SQL- Elemente de bază

- Tipuri compuse:

RECORD

TABLE

VARRAY

- Tipuri referință:

REF CURSOR

REF tip-obiect

Tipuri LOB:

BFILE

BLOB

CLOB

NCLOB

# Limbajul PL/SQL- Elemente de bază

- `BINARY_INTEGER` : $[-2^{31}, 2^{31}-1]$

Subtipuri ale lui `BINARY_INTEGER`:

-`SUBTYPE` numesubtip `IS` tip-baza

Subtipuri predefinite:

-`NATURAL`

-`NATURALN`

-`POSITIVE`

-`POSITIVEN`

-`SIGNTYPE`

# Limbajul PL/SQL- Elemente de bază

- NUMBER[(precizie,nrzec)], 1.0E-130.. 9.99E125
- NUMBER(precizie) max precizie – 38  
nrzec ∈ [-84,127]

Subtipuri ale lui NUMBER:

-DEC

INTEGER

-DECIMAL

INT

-NUMERIC

DOUBLE PRECISION

-FLOAT

REAL

-SMALLINT

- PLS\_INTEGER

# Limbajul PL/SQL- Elemente de bază

- Tipuri caracter:
  - CHAR[(lungimef)], lungimef<=32767octeti  
max coloana tabela CHAR =2000 octeti
  - LONG lungime max=32760  
max coloana tabela LONG=2\*\*31-1 octeti

Coloanele de tip LONG utilizate in: SELECT, UPDATE, INSERT dar nu in expresii, functii SQL, clauze WHERE, GROUP BY, CONNECT BY

-

# Limbajul PL/SQL- Elemente de bază

RAW(lungime), lungime $\leq$ 32767.

max lungime coloana tabela =2000 octeti.

- LONG RAW lungime max.=32760.

max lungime coloana tabela LONG RAW  $2^{31}-1$ .

- ROWID - identificator de linie
  - numar obiect date(nr.segment date)
  - numar fisier date
  - numar bloc date in fisier date
  - numar linie in bloc de date

VARCHAR2(lungime) lungime  $\leq$ 32767

Subtipuri: STRING, VARCHAR

# Limbajul PL/SQL- Elemente de bază

- National Language Support(NLS).
  - multime caractere database
  - multime caractere nationale

NCHAR[(lungime)], lungime<=32767.

coduri US7ASCII, JA16SJIS. Lungime in caractere.

max lungime coloana tabela NCHAR=2000

NVARCHAR2(lungimemax), <=32767

max lungime coloana tabela =4000

# Limbajul PL/SQL- Elemente de bază

- Tipuri LOB(Large Objects) 4Go

Mem.valori numite locatori.

Pentru coloanele tip BLOB, CLOB,NCLOB datele memorate in baza de date. Pentru BFILE datele in fisiere ale SO.

- BFILE. nr. max SESSION\_MAX-OPEN\_FILES  
. Read Only
- BLOB Participă in tranzactii
- CLOB blocuri de caractere
- NCLOB blocuri 1 octet sau multiocteti.

# Limbajul PL/SQL- Elemente de bază

- Alte tipuri:
  - BOOLEAN -operatii logice
  - DATE. functia SYSDATE return data si timpul.  
Jan.1 4712 BC.. December 31, 4712 AD  
O data Julian =nr.zile din 1ian.4712  
Functiile TO\_DATE , TO\_CHAR cu format 'J', da conversia valori DATE si echivalentul Julian.  
Formatul implicit dată este dat de parametrul NLS\_DATE\_FORMAT.  
Ex. 'DD-MON-YY'

# Limbajul PL/SQL- Elemente de bază

- Definierea tipurilor
  - Tipuri de baza(BINARY\_INTEGER, DEC..)
  - Tipuri definite de utilizator:  
TYPE numetip IS ...
- Definierea variabilelor:  
numevariabila numetip[:=expresie];  
numevariabila numesubtip[:=expresie];

# Limbaajul PL/SQL- Elemente de bază

- Definierea subtipurilor:
  - SUBTYPE numesubtip IS tipbaza;
  - SUBTYPE numesubtip IS tabela.cimp%TYPE;
  - SUBTYPE numesubtip IS variabila%TYPE;
  - SUBTYPE numesubtip IS cursor%ROWTYPE

# Limbaajul PL/SQL- Elemente de bază

DECLARE

SUBTYPE EmpDate IS DATE;

SUBTYPE alfa IS NATURAL;

TYPE Nume1 IS TABLE OF VARCHAR2(15);

SUBTYPE beta IS nume1;

TYPE timprec IS RECORD( minute INTEGER,  
ore INTEGER);

SUBTYPE timp IS timprec;

# Limbaajul PL/SQL- Elemente de bază

```
SUBTYPE numeid IS emp.empno%TYPE;
```

```
CURSOR c1 IS SELECT * FROM dept;
```

```
SUBTYPE deptlinie IS c1%ROWTYPE;
```

In definiția unui subtip nu se poate specifica restricție pe tipul de baza.

```
SUBTYPE gama1 IS NUMBER(6,2);
```

```
SUBTYPE gama2 IS CHAR(1);
```

```
SUBTYPE gama3 IS VARCHAR(10);
```

# Limbajul PL/SQL- Elemente de bază

- Definirea de restrictii indirect:

```
DECLARE
```

```
var1 VARCHAR2(10);
```

```
SUBTYPE vartip IS var1%TYPE;
```

- Folosirea subtipurilor:

```
DECLARE
```

```
SUBTYPE contor IS NATURAL;
```

```
nr1 contor;
```

```
nr2 contor;
```

# Limbajul PL/SQL- Elemente de bază

- DECLARE  
SUBTYPE alfatip IS NUMBER;  
total alfatip(8,2);
- DECLARE  
temp NUMBER(2,0);  
SUBTYPE alfa IS temp%TYPE;  
x alfa;  
y alfa;  
BEGIN  
x:=100; -- eroare

# Limbajul PL/SQL- Elemente de bază

- Compatibilitatea tipurilor:

Un subtip nerestrins este interschimbabil cu tipul de bază.

```
DECLARE
```

```
    SUBTYPE tipx IS NUMBER;
```

```
    cant NUMBER(8,2);
```

```
    total tipx;
```

```
BEGIN
```

```
    total:=cant;
```

# Limbajul PL/SQL- Elemente de bază

- Subtipuri diferite sunt interschimbabile dacă au același tip de bază.

```
DECLARE
```

```
    SUBTYPE tip1 IS BOOLEAN;
```

```
    SUBTYPE tip2 IS BOOLEAN;
```

```
    alfa tip1;
```

```
    beta tip2;
```

```
BEGIN
```

```
    alfa:=beta;
```

# Limbajul PL/SQL- Elemente de bază

- Subtipuri diferite sunt interschimbabile dacă tipurile lor de bază sunt in aceeași familie.

```
DECLARE
```

```
    SUBTYPE tip1 IS CHAR;
```

```
    SUBTYPE tip2 IS VARCHAR2;
```

```
    alfa tip1;
```

```
    beta tip2;
```

```
BEGIN
```

```
    alfa:=beta;
```

# Limbajul PL/SQL- Elemente de bază

- Conversia tipurilor de date.

## 1. Conversia explicită:

TO\_DATE de la CHAR la DATE

TO\_NUMBER de la CHAR la NUMBER

TO\_CHAR de la DATE sau NUMBER la CHAR

## 2. Conversia implicită:

DECLARE

timp1 char(5);

timp2 char(5);

# Limbajul PL/SQL- Elemente de bază

```
diferenta number(5);  
BEGIN  
SELECT TO_CHAR(SYSDATE,'SSSSS')  
INTO timp1 FROM sys.dual;  
    secventa comenzi  
SELECT TO_CHAR(SYSDATE,'SSSSS')  
INTO timp2 FROM sys.dual;  
    diferenta:=timp2-timp1;  
INSERT INTO rezultat VALUES (diferenta,..)  
END;
```

# Limbajul PL/SQL- Elemente de bază

---

	bin_int	char	date	long	number	pls_int	raw	rowid	vvarchar2
bin_int		*		*	*	*			*
Char	*		*	*					*
Date		*		*					*
Long		*					*		*
Number	*	*		*		*			*
Pls_int	*	*		*	*				*
Raw		*		*					*
Rowid		*							*
Vvarchar2	*	*	*	*	*	*	*	*	

---

# Limbajul PL/SQL- Elemente de bază

Ex. '02-JUL-99' -> DATE

'xyDTRA' -> DATE ?

'12345ABCD' -> NUMBER ?

- RAW sau LONG RAW in CHAR sau VARCHAR2. Coloana->variabilă  
octet-> 2 caractere cd, unde c si d sunt cifre hexazecimale.
- CHAR sau VARCHAR2 in RAW sau LONG RAW. Variabilă->coloană

# Limbaajul PL/SQL- Elemente de bază

- Declaraarea variabilelor și constantelor:  
identificator tip [:=expresie];  
identificator CONSTANT tip:=expresie;  
variabile neinitializate – NULL
- Cuvintul DEFAULT ptr.inițializarea variabilelor  
alfa BOOLEAN DEFAULT FALSE;
- NOT NULL in declarații urmată de inițializare.

# Limbajul PL/SQL- Elemente de bază

- Folosirea %TYPE:

Specifică tipul de dată al variabilelor sau coloanelor. NOT NULL nu se transmite.

```
DECLARE
```

```
    nume emp.empno%TYPE
```

```
BEGIN
```

```
    nume:=NULL;
```

# Limbajul PL/SQL- Elemente de bază

- Utilizare %ROWTYPE:

Tip de înregistrare – o linie tabelă sau view

DECLARE

```
emp_rec emp%TYPE;
```

```
CURSOR c1 IS SELECT deptno,dname,loc  
FROM dept;
```

```
dept_rec c1%ROWTYPE;
```

Coloanele din tip înregistrare au același tip cu câmpurile din înregistrare.

```
SELECT * INTO emp_rec FROM emp where...
```

# Limbajul PL/SQL- Elemente de bază

- Valorile coloanelor returnate de SELECT -> câmpurile înregistrării.
- Referința la un câmp: inregistrare.câmp
- Asignare agregată:

Declarația cu %ROWTYPE nu are clauză de iniț.

```
DECLARE
```

```
dept_rec1 dept%ROWTYPE;
```

```
dept_rec2 dept%ROWTYPE;
```

```
CURSOR c1 IS SELECT deptno,dname,loc  
FROM dept;
```

# Limbajul PL/SQL- Elemente de bază

```
dept_rec3 c1%ROWTYPE;
```

```
BEGIN
```

```
...
```

```
dept_rec1:=dept_rec2;
```

```
dept_rec2:=dept_rec3; --incorect
```

Putem asigna o lista de valori la o inregistrare cu

SELECT sau FETCH:

```
SELECT deptno,dname,loc INTO dept_rec1  
FROM dept WHERE deptno=10;
```

# Limbajul PL/SQL- Elemente de bază

- Nu putem asigna o lista de valori pentru o asignare:

Numeinregistrare:=(val1,val2,...)—incorect

- Nu putem folosi inregistrarile pentru inserare sau actualizare:

INSERT INTO dept VALUES  
(numeinregistrare)—incorect

PL/SQL nu permite referințe forward

- Nu putem defini același tip de data pentru mai multe variabile.

# Limbajul PL/SQL- Elemente de bază

- Precedența operatorilor:

**\*\***, **NOT**

**+**, **-**

**\***, **/**

**+**, **-**, **||**

**=**, **!=**, **<**, **>**, **<=**, **>=**

**IS NULL**, **LIKE**, **BETWEEN**, **IN**

**AND**

**OR**

# Limbajul PL/SQL- Elemente de bază

- Funcția NVL

NVL(exp1,exp2) returnează exp2 dacă exp1=NULL  
altfel exp1

- Funcția REPLACE

REPLACE(sir1,sir2,sir3) returnează:

sir1 dacă sir2 este NULL, indiferent de sir3

sir1 din care se elimină toate aparițiile lui sir2, cind sir2 nu e NULL si sir3 nu e NULL

sir1 cind sir2 este NOT NULL si sir3 este NULL

# Limbaajul PL/SQL- Elemente de bază

- Funcții “Built-in”

1) privitoare la erori: SQLCODE, SQLERRM

2) privitoare la numere:

ABS	COSH	ROUND	
ACOS	EXP	SIGN	
ASIN	FLOOR	SIN	
ATAN	LN	SINH	
ATAN2	LOG	SQRT	
CEIL	MOD	TAN	
COS	POWER	TANH	TRUNC <sub>41</sub>

# Limbaajul PL/SQL- Elemente de bază

- 3)Funcții caracter:

ASCII

LPAD

RTRIM

CHR

LTRIM

SOUNDEX

CONCAT

NLS\_INITCAP

SUBSTR

INITCAP

NLS\_LOWER

SUBSTRB

INSTR

NLS\_UPPER

TRANSLATE

INSTRB

NLSSORT

UPPER

LENGTH

REPLACE

LOWER

RPAD

# Limbajul PL/SQL- Elemente de bază

- 4) Conversii:

CHARTOROWID	TO_CHAR
CONVERT	TO_DATE
HEXTORAW	TO_LABEL
NLS_CHARSET_ID	TO_MULTI_BYTE
NLS_CHARSET_NAME	TO_NUMBER
RAWTOHEX	TO_SINGLE_BYTE
ROWIDTOCHAR	

# Limbaajul PL/SQL- Elemente de bază

- 5) Dată calendaristică:

ADD\_MONTHS

LAST\_DAY

MONTHS\_BETWEEN

NEW\_TIME

NEXT\_DAY

ROUND

SYSDATE

TRUNC

# Limbajul PL/SQL- Elemente de bază

- 6) Diverse:

DECODE

LEAST\_UB

DUMP

NVL

GREATEST

UID

GREATEST\_LB

USER

LEAST

USERENV

VSIZE

# Limbajul PL/SQL- Structuri de control

- IF-THEN

```
IF conditie THEN  
    sir de comenzi;  
END IF;
```

- IF-THEN-ELSE:

```
IF conditie THEN  
    sir de comenzi1;  
ELSE  
    sir de comenzi2;  
END IF;
```

# Limbajul PL/SQL- Structuri de control

- IF-THEN-ELSIF

```
IF conditi1 THEN
    sir de comenzi1;
ELSIF conditie2 THEN
    sir de comenzi2;
ELSE
    sir de comenzi3;
END IF;
```

# Limbajul PL/SQL- Structuri de control

- LOOP  
LOOP  
sir de comenzi;  
END LOOP;
- EXIT si EXIT-WHEN  
LOOP  
...  
IF conditie THEN  
EXIT;  
END IF  
END LOOP;

# Limbajul PL/SQL- Structuri de control

- LOOP

```
FETCH c1 INTO ...
```

```
EXIT WHEN c1%NOTFOUND;
```

```
....
```

```
END LOOP;
```

```
CLOSE c1;
```

```
IF c1%NOTFOUND THEN
```

```
    EXIT;
```

```
END IF;
```

# Limbajul PL/SQL- Structuri de control

- Etichete LOOP

<<nume eticheta>>

LOOP

  sir de comenzi;

END LOOP [nume eticheta];

- <<e1>>

LOOP

...

<<e2>>

# Limbajul PL/SQL- Structuri de control

...

```
EXIT e1 WHEN conditie;  
END LOOP e2;
```

.....

```
END LOOP e1;
```

- WHILE-LOOP

```
WHILE conditie LOOP  
  sir de comenzi;  
END LOOP;
```

# Limbajul PL/SQL- Structuri de control

- FOR-LOOP

```
FOR contor IN [REVERSE] min..max LOOP  
    sir de comenzi;  
END LOOP;
```

In interior contor este referit ca o constantă.

min, max pot fi :literali, variabile, expresii evaluate ca intregi.

```
STEP h - IF MOD(contor,h)=0 THEN ...
```

contor nu trebuie definit.

# Limbajul PL/SQL- Structuri de control

Ex. DECLARE

contor integer;

BEGIN

FOR contor IN 1.. 20 LOOP

....

IF contor >8 THEN ... --referă contor din FOR

END IF;

END LOOP;

END;

# Limbajul PL/SQL- Structuri de control

- Pentru a referi variabila globală contor vom da:

<<et1>>

DECLARE

contor INTEGER;

BEGIN

FOR contor IN 1..20 LOOP

IF et1.contor > 8 THEN ...

END IF;

END LOOP;

END et1;

Același lucru și pentru FOR imbricați.

# Limbajul PL/SQL- Structuri de control

<<et1>>

```
FOR pas IN 1..20 LOOP
```

```
  FOR pas IN 1..5 LOOP
```

```
    ....
```

```
    IF et1.pas >=12 THEN ...
```

```
  END IF;
```

```
END LOOP;
```

```
END LOOP et1;
```

# Limbajul PL/SQL- Structuri de control

- Folosirea lui EXIT:

```
DECLARE
```

```
emp_rec emp%ROWTYPE;
```

```
CURSOR c1 IS SELECT deptno,dname,loc  
FROM dept;
```

```
numarrec integer :=0;
```

```
BEGIN
```

```
SELECT COUNT(*) INTO numarrec FROM c1;
```

```
FOR j IN 1..numarrec LOOP
```

# Limbajul PL/SQL- Structuri de control

```
FETCH c1 INTO emp_rec;
```

```
EXIT WHEN c1%NOTFOUND;
```

```
....
```

```
END LOOP;
```

```
END;
```

# Limbajul PL/SQL- Structuri de control

```
<<et1>>
```

```
FOR i IN 1..10 LOOP
```

```
...
```

```
FOR j IN 1..5 LOOP
```

```
  FETCH c1 INTO emp_rec;
```

```
  EXIT et1 WHEN c1%NOTFOUND;
```

```
....
```

```
  END LOOP;
```

```
END LOOP et1;
```

```
--aici trece controlul
```

# Limbajul PL/SQL- Structuri de control

- Instrucțiunile GOTO și NULL.

-GOTO eticheta

După eticheta trebuie sa fie o instrucțiune executabilă

END LOOP nu este executabilă. Folosim NULL;

GOTO nu poate face saltul in interiorul unui IF, instrucțiune LOOP sau subbloc.

GOTO nu poate face saltul de la o clauză IF la alta clauză a aceluși IF.

GOTO nu poate face saltul dintr-un bloc de exceptii în blocul curent.

# Limbajul PL/SQL- Colecții și înregistrări

- Colecție=grup ordonat de elemente de același tip.
- Un element are indice unic în colecție.
- Două tipuri de colecții: tabele nested și tablouri de dimensiune variabilă-varrays.
- Colecțiile pot memora instanțe ale unui tip obiect.
- Colecțiile pot fi attribute ale unui tip obiect.
- Colecțiile pot fi folosite ca parametri.
- Tabelele nested pot fi interpretate ca tabele cu o singură coloană.

# Limbaajul PL/SQL- Colectii și înregistrări

- Arrays au un număr maxim de elemente.
- Tabelele nested sunt nemărginite.
- Arrays trebuie să fie dense, fără spații între elemente.
- Nu se pot șterge elemente individuale din array.
- Inițial tabelele nested sunt dense.
- Cu procedura DELETE se pot șterge elemente în tabele nested.
- Cu procedura NEXT operăm asupra indicilor.

# Limbaajul PL/SQL- Colectii și înregistrări

- Diferențe între tabele nested și tabele index-by.
  - Tipuri permise pentru tabele index-by dar nu pentru tabele nested:  
Binary\_integer, Boolean, Long, Long Raw, Natural, NaturalN, Pls\_integer, Positive, PositiveN, Sigttype, String.
  - Tabelele index-by sunt definite cu clauza INDEX BY BINARY\_INTEGER.
  - O tabelă nested neinițializată este NULL, însă o tabelă index-by neinițializată este vidă.

# Limbaajul PL/SQL- Colectii și înregistrări

- Pentru tabele nested indicele  $1..2^{*31}-1$
- Pentru tabele index-by indicele  $-2^{**31}.. 2^{**31}-1$
- Pentru extinderea unei tabele nested folosim procedura EXTEND, pentru tabele index-by specificăm indici mai mari.
- Procedurile EXTEND și TRIM se pot aplica numai la tabele nested.

# Limbajul PL/SQL- Colecții și înregistrări

- Definirea și declararea colecțiilor

TYPE numetip IS TABLE OF tipelement [NOT NULL]

TYPE numetip IS {VARRAY|VARYING ARRAY}(max) OF tipelement [NOT NULL]

Tipelement este orice tip PL/SQL cu excepția tipurilor:

BINARY\_INTEGER NCLOB SIGNTYPE

BOOLEAN NVARCHAR2 STRING

LONG PLS\_INTEGER TABLE

LONG RAW POSITIVE VARRAY

NATURAL POSITIVEN

NATURALN REF CURSOR

NCHAR Tipuri obiect cu attribute TABLE sau VARRAY

# Limbaajul PL/SQL- Colectii și înregistrări

- Dacă tipelement este tip RECORD, atunci un câmp din înregistrare trebuie să fie tip scalar sau tip obiect.
- Definierea tipului tabele index-by:

TYPE numetip IS TABLE OF tipelement [NOT NULL] INDEXED BY BINARY\_INTEGER;

Tipuri permise pentru tabele index-by:

BINARY_INTEGER	NATURAL	POSITIVEN
BOOLEAN	NATURALN	SIGNTYPE
LONG	PLS_INTEGER	STRING
LONG RAW	POSITIVE	

# Limbajul PL/SQL- Colecții și înregistrări

Tabelele index-by initial nu sunt dense. Ex.

```
DECLARE
```

```
TYPE tip1 IS TABLE OF emp%ROWTYPE  
INDEX BY BINARY_INTEGER;
```

```
emp_tab_by tip1;
```

```
BEGIN
```

```
SELECT * INTO emp_tab_by(100) FROM emp  
WHERE empno=100;
```

```
END;
```

## Limbajul PL/SQL- Colecții și înregistrări

- Folosim %TYPE ce dă tipul unei variabile sau coloană a unei tabele.
- Folosim %ROWTYPE ce dă tipul de linie a unui cursor sau tabelă.

DECLARE

```
TYPE lista_emp IS TABLE OF emp.ename%TYPE;
```

```
CURSOR c1 IS SELECT * FROM dept;
```

```
TYPE tip_dept IS VARRAY(15) OF c1%ROWTYPE;
```

```
BEGIN
```

```
....
```

# Limbajul PL/SQL- Colecții și înregistrări

DECLARE

```
TYPE inreg1 IS RECORD (  
    nume VARCHAR2(20),  
    nrcopii NATURAL);
```

```
TYPE persoane IS VARRAY(100) OF inreg1;
```

BEGIN

...

# Limbajul PL/SQL- Colecții și înregistrări

- Declaraarea colecțiilor

identificator numetip;

```
CREATE TYPE lista_cursuri AS TABLE OF  
VARCHAR2(10)
```

```
/
```

```
CREATE TYPE Student AS OBJECT (  
marca INTEGER(4),  
nume VARCHAR2(15),  
cursuri lista_cursuri)
```

```
/
```

# Limbajul PL/SQL- Colecții și înregistrări

```
CREATE TYPE Proiect AS OBJECT(  
  nr_proiect NUMBER(2),  
  cost      NUMBER(7,2))  
/  
CREATE TYPE lista_proiecte AS VARRAY(20) OF  
  proiect  
/  
CREATE TABLE depart2 (  
  id_dept NUMBER(2),  
  nume    VARCHAR2(20),  
  proiecte lista_proiecte)  
/  

```