SGBD - curs 1

4.10.2011

Etapele proiectării unui sistem informatic

1. Descrierea sistemului real
2. Restricțiile de funcționare ale modelului (constrângeri)
3. Entitățile modelului și atributele acestora
4. Relațiile modelului și cardinalitățile acestuia
5. Diagrama E-R
6. Diagrama conceptuală (∧, △)
7. Schemele relaționale
8. Normalizarea / denormalizarea modelului
9. Sintaxa SQL pt crearea tabelelor
10. Ce se dorește în continuare de la model)

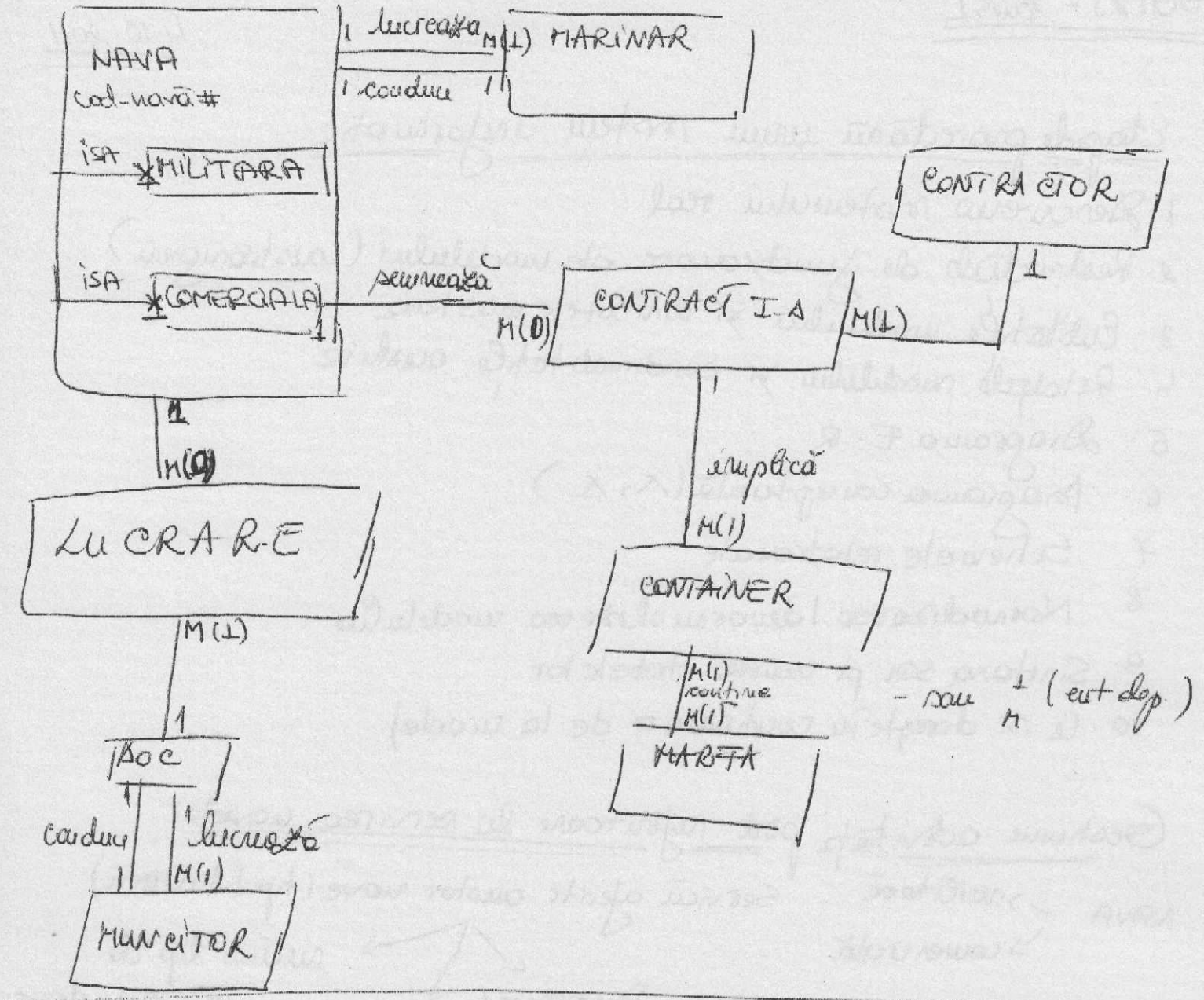
Gestiune activități port referitoare la servirea navelor

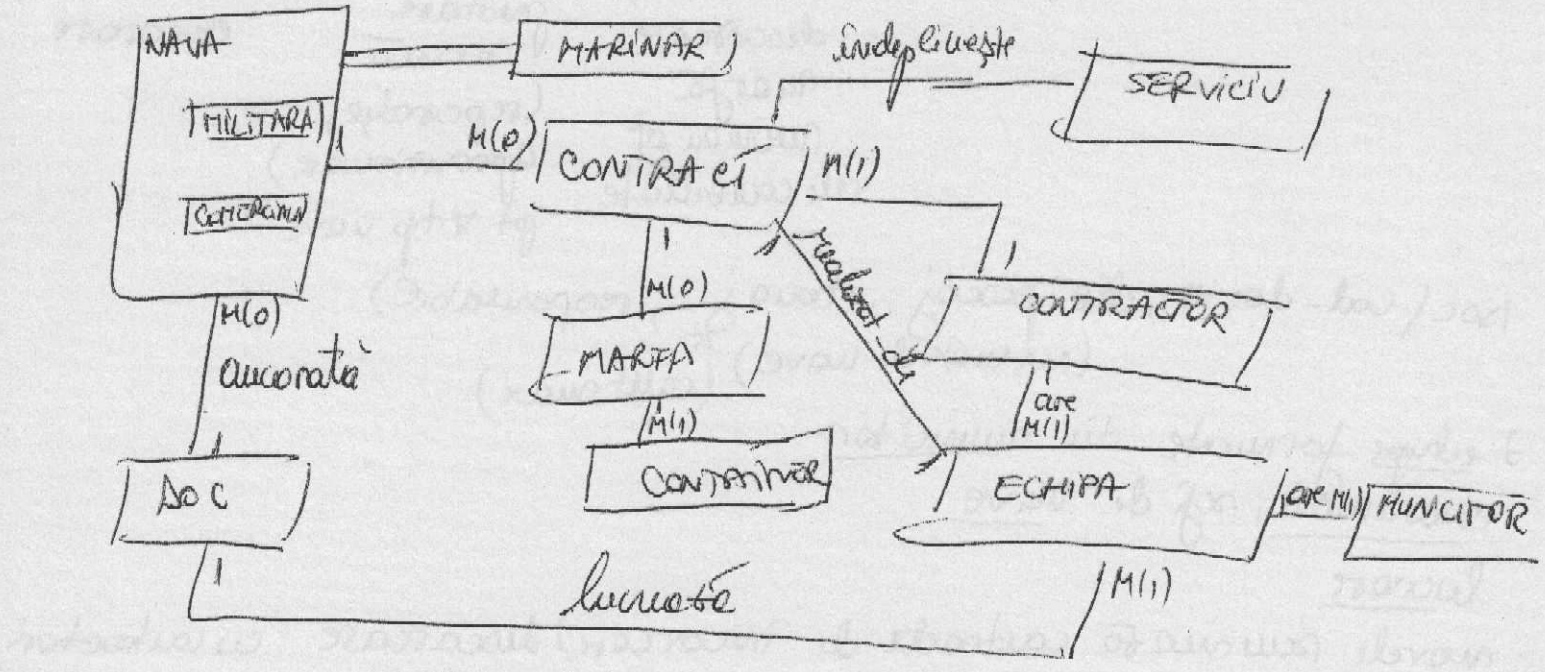
NAVA → militară
→ comercială

Servicii oferite acestor nave (tipul lucrării)
→ încărcare / descărcare marfă numai pt cele comerciale
→ prestare servicii (reparație, aprovizionare) pt ∀ tip nave
→ niciun tip de serviciu - doar ancorare

DOC (cod-doc#, ~~dim~~ pescaj, tonaj, responsabil)
(nr max de nave) (cant max)

7 echipe formate din muncitori
marinari, șefi de nave
lucrare
navele semnează contracte de încărcare / descărcare cu contractori
marfa vine într-un container
~~(numai portocale)~~





Temă: Care sunt corecte? Corectați

Gestiunea unei linii de producție (mezeluri, pantofi)
(materie primă, rețete) produse
↑
piele, carne

BD – Curs 2

11.10.2011

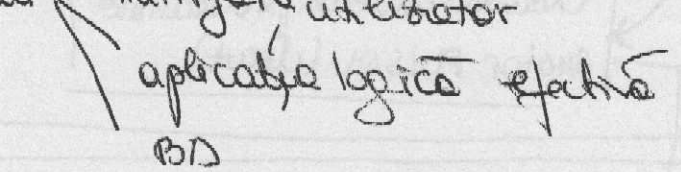
PL/SQL – este un limbaj de programare care
1. asigură accesarea informației dintr-o BD relațională OO.
2. permite gruparea cererilor într-un bloc unic de tratare a datelor

Problematici abordate

1. Concepte generale
2. Controlul execuției unui program
3. Tipuri de date
4. Cursoare
5. Subprograme
6. Pachete / biblioteci
7. SQL dinamic
8. Declanșatori
9. Gestiunea și tratarea erorilor.

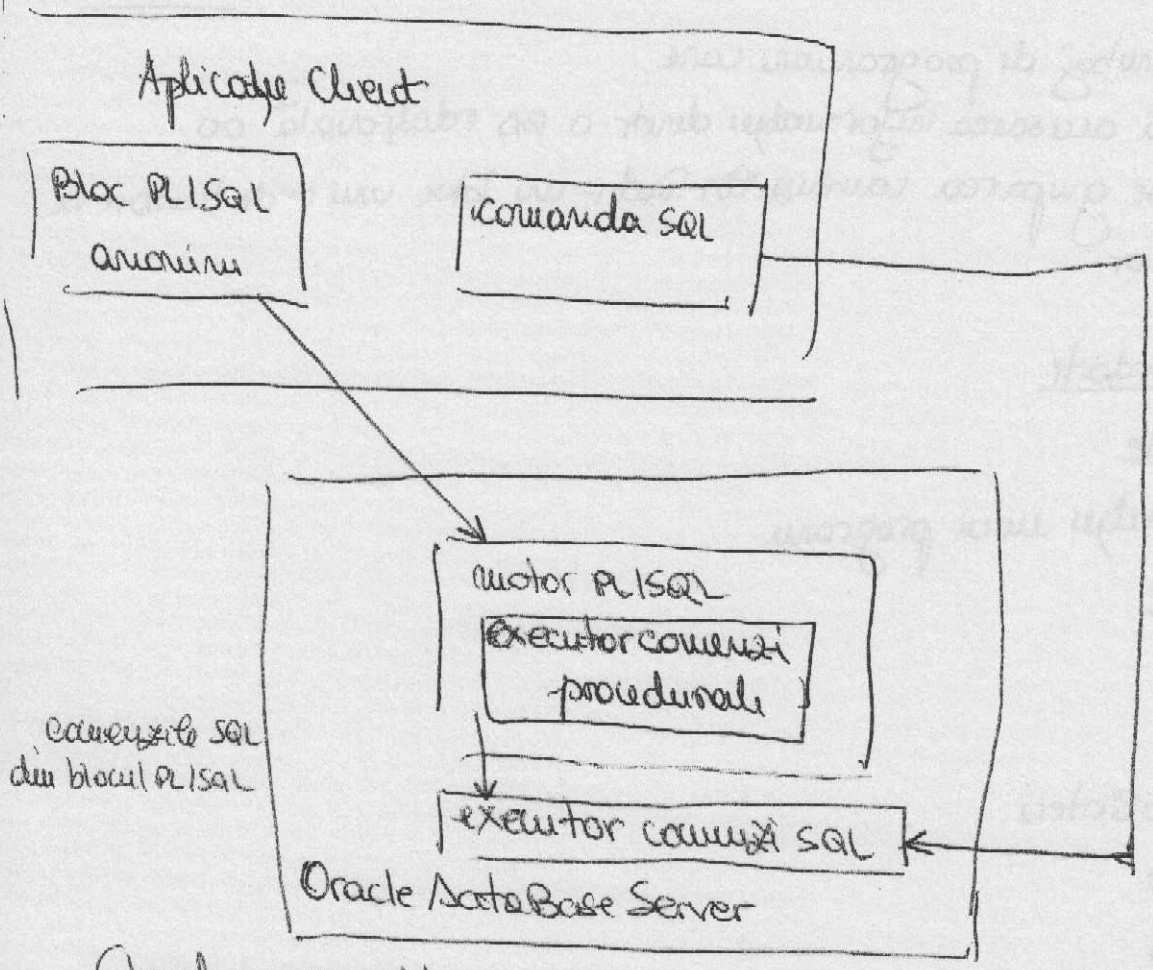
Concepte generale

Blocurile PL/SQL sunt transmise unui motor PL/SQL și procesate (compilate, executate) de acesta. Motorul PL/SQL poate să se afle în serverul Oracle sau într-un utilitar iar utilizarea se deosebește de unde se invocă. Blocurile PL/SQL pot fi executate pe stație client fără interacțiune cu serverul sau în întregime pe server. O aplicație BD poate fi structurată în:



Există 2 modele pentru proiectarea unei aplicații BD
- modelul client-server
- modelul three-tier

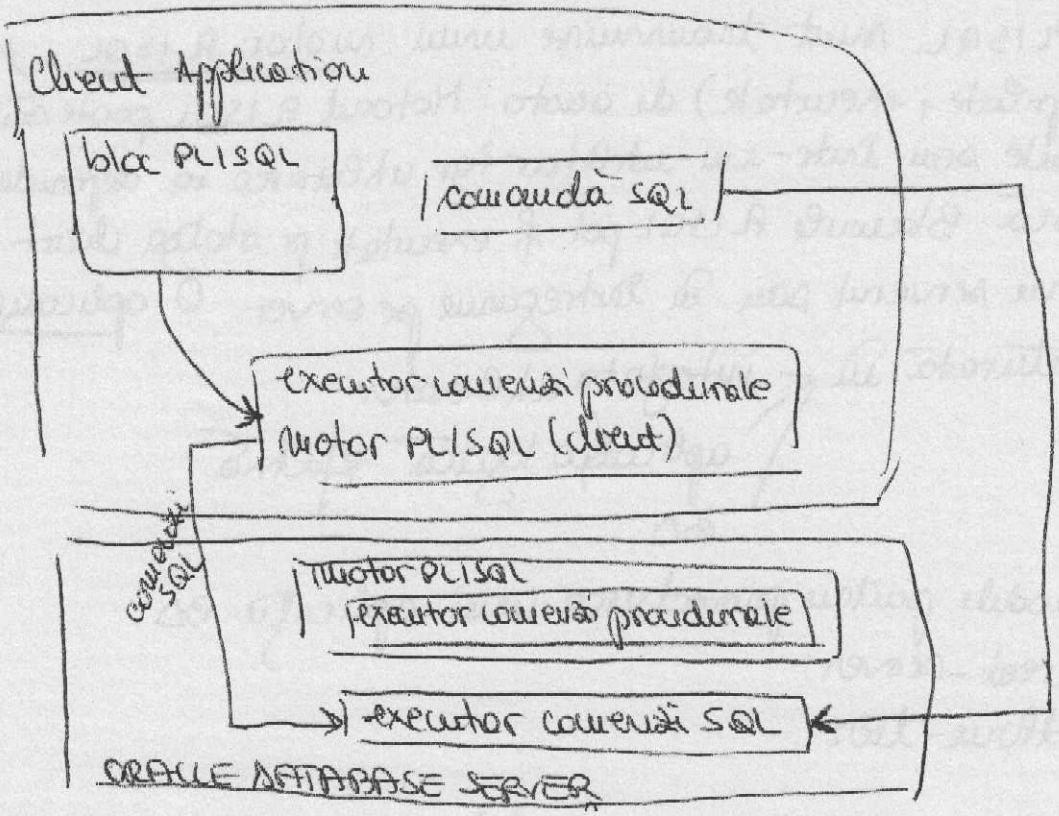
Motor PL/SQL server side



Structura permite executarea de cod PL/SQL stocat în BD

Motor PL/SQL client side

- care permite executarea de cod PL/SQL în ORACLE FORMS & REPORT pt a genera forme & rapoarte



Daca aplicatia are subprograme stocate, atunci este utilizat motorul PL/SQL de pe server

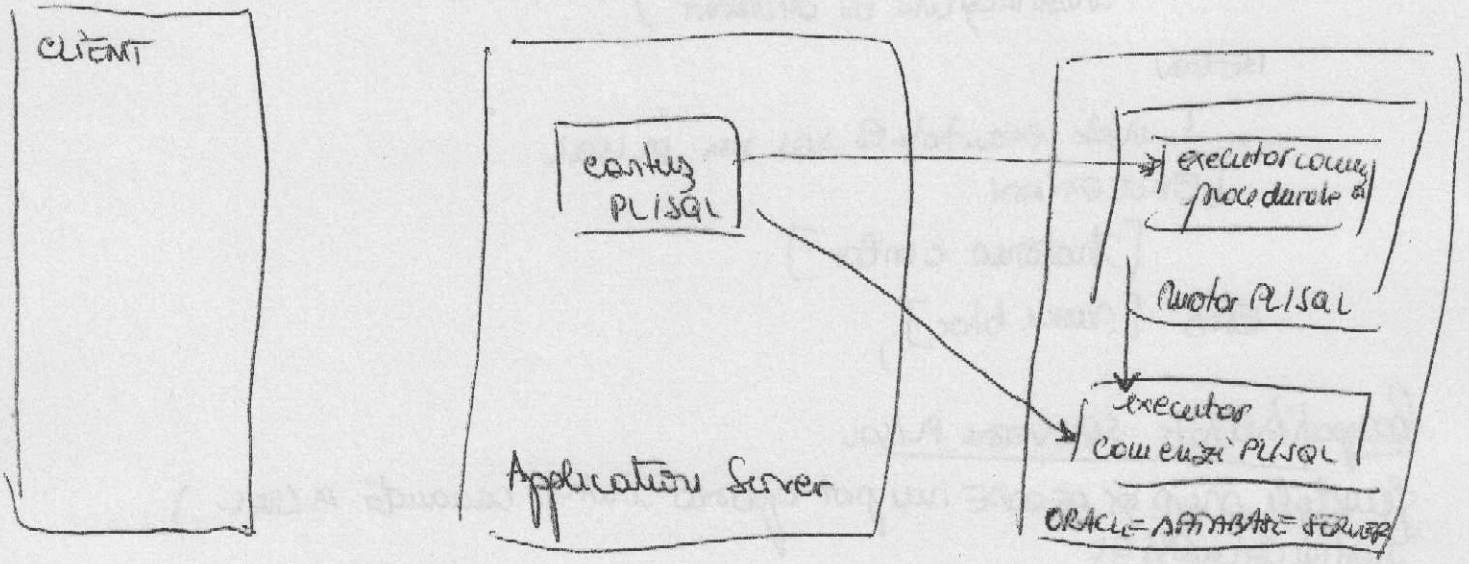
-Consideram un scenariu in care 2 motoare PL/SQL: unul local pe statia client si unul pe server

Ex. un trigger e se executa pe statia client si care apeleaza o procedura stocata in BD.

3. Modelul three-tier (client server pe 3 nivele)

-aplicatia logica a BD este impartita in 3 parti

- client (browser)
- server de aplicatie
- server baza de date



Codul va contine ca si client pt server

Comenzile SQL si blocurile PL/SQL in general sunt trimise serverului pt a fi executate. Pt a realiza acest lucru, trb. stabilita conectarea la BD. PL/SQL nu permite nici o sintaxa care sa permita aceasta conectare.
cadrul

Pt ce fel de aplicatii este necesar PL/SQL?

1) proceduri si functii stocate
2) pachetele
3) Declansatori BD
4) Application Trigger

Controlul execuției unui program

Un program PL/SQL poate conține unul sau mai multe blocuri. Un bloc poate fi:
- anonim
- nenumit – blocuri etichetate, construite static sau dinamic și executate o singură dată
  - subprograme
  - pachete
  - declanșatori

Structura bloc SQL

```
[<< nume bloc >>]
[DECLARE
    instrucțiuni de declarare ]
BEGIN
    } instr. executabile SQL sau PL/SQL
[EXCEPTION
    [tratarea erorilor ]
END [nume bloc];
```

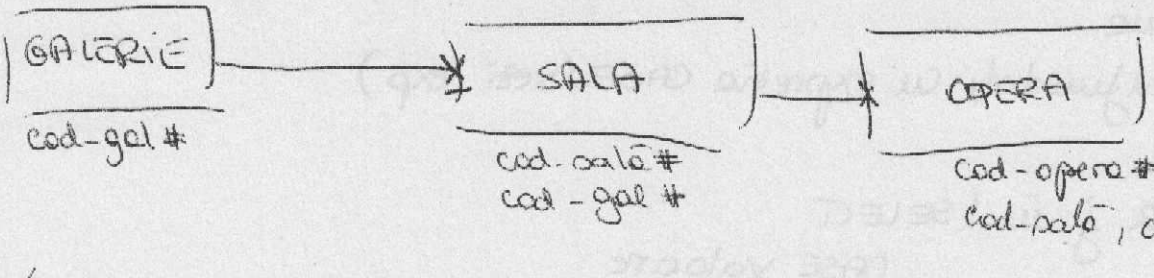
Incompatibilități SQL versus PL/SQL

(Funcțiile group by și DECODE nu pot apărea într-o comandă PL/SQL)

Instrucțiunile SQL
1. comentarii PL/SQL
2. operare
3. variabile globale
4. comenzi SQL *Plus ∈ {bloc PL/SQL}
5. Blocul PL/SQL este o unitate tranzacțională? NU
6. PL/SQL nu suportă comenzile SQL prin care se acordă/revocă privilegii GRANT; REVOKE
7. SELECT → nu întoarce nicio rezultat NO_DATA_FOUND
   → întoarce mai multe linii TOO_MANY_ROWS
   → întoarce o singură linie

clauza INTO în comanda select

SELECT
INTO (a, b, c)
FROM
(:x (variabile de legătură)



Să se specifice dacă o galerie este mare medie sau mică după cum un operator de artă e > 200, între 100 – 200, < 100.

SET SERVEROUTPUT ON ⟶ permite activarea pachetului DBMS_OUTPUT,

DEFINE p_cod_gal = 753

DECLARE

~~x nume_tabel.nume~~ x nume_tabel.nume_camp %TYPE;

v_cod_galerie opera.cod-galerie%TYPE := &p_cod_gal (continuat)

v_numar NUMBER(3) := 0;

v_comentariu VARCHAR2(10); aici zic mică, mare, etc.

BEGIN

SELECT COUNT(*)

INTO v_numar

FROM OPERA

WHERE cod_galerie = v_cod_galerie;

IF v_numar < 100 THEN

v_comentariu := 'mică';

ELSIF v_numar BETWEEN 100 AND 200 THEN

v_comentariu := 'mediu';

ELSE

v_comentariu := 'mare';

END IF;

DBMS_OUTPUT.PUT_LINE ('Galeria având codul ' || v_cod_galerie ||

' este de tip ' || v_comentariu);

END;

/ ⟶ încarcă și execută

SET SERVEROUTPUT OFF

CASE - 2 forme
- nu confundați cu expresia CASE (vezi exp)

Exp. BEGIN
FOR j IN (SELECT
CASE valoare
WHEN 1000 THEN 1100
WHEN 10000 THEN 11000
WHEN 100000 THEN 110000
ELSE valoare
END
from opera)
END LOOP;
END;

4) Instrucțiuni iterative → LOOP → are obligatoriu condiție de ieșire din buclă
→ WHILE
→ FOR

LOOP
secvență instr.
END LOOP;

Pe structura tabelului OPERA se va introduce un nou câmp (stea) fără să mai facă un bloc PL/SQL care va introduce 0 * pt fiecare 10000 $ din val unei opere al cărei cod e specificat

ALTER TABLE opera
ADD stea VARCHAR2(20);

DEFINE p_cod_opera = 7772
DECLARE
v_cod_opera opera.cod_opera%TYPE := &p_cod_opera;
v_valoare opera.valoare%TYPE;
v_stea opera.stea%TYPE := NULL;
BEGIN
SELECT NVL(ROUND(valoare/10000),0)
INTO v_valoare
from opera
where cod_opera = v_cod_opera;

```
if v_valoare>0 then
    for i in 1..v_valoare LOOP
        v_stea:=v_stea||'*';
    END LOOP;
END IF;
UPDATE opera
 SET stea=v_stea
 where cod_opera=v_cod_opera;
 COMMIT;
END;
```

Instr de salt: EXIT -> iesire dintr-un ciclu

```
EXIT [nume_eticheta] [WHEN conditie];
```

Instr vida (NULL)

```
if (a=b) then
        NULL;
 else
     DBMS_OUTPUT.PUT_LINE('a<>b');
 END IF;
```

Instr GOTO

SGBD curs 3

18.10.2011

Tipuri de date în PL/SQL

Prin tip de date se specifică → formatul de stocare
→ constrângerile care trebuie verif
→ domeniul valorilor sale

Var PL/SQL → specific PL/SQL
→ nespecific PL/SQL

Nespecific

- de tip scalar → numerice
  → caracter
  → logice / booleene (true, false, null)
  → dată calendaristică
  → globalizare (NCHAR, NVARCHAR2)
- compuse → record
  → colecții → tabele indexate
  → tabele imbricate (nested table)
  → vectori
- LOB → interne (CLOB, BLOB, NCLOB) depuse în BD
  (CLOB caract, BLOB poze, depuse)
  → externe (BFILE) → în afara BD
  (file)
- tip referință → REF cursor
  → REF obiect
- tip obiect

Tipuri nespecifice PL/SQL → var de legătură (bind)
→ var gazdă
→ var indicator

Tipuri de date globalizare

PL/SQL suportă 2 seturi de caractere → 1) specifică BD-ului care este utilizat pt definirea identificatorilor și a codului sursă
2) o mulțime de caractere naționale folosite pt reprezentarea datelor cu caracter național (NCS)

Tipurile NCHAR, NVARCHAR permit stocarea în BD a șirurilor de caractere ce folosesc multiplu NLS. Aceste tipuri de date suportă numai date unicode. Unicode furnizează o valoare cod unică pt fiecare caracter indiferent de program, platformă sau limbă.

LOB

sunt tipuri de date ale căror valori numite locatori specifică localizarea unor obiecte de dimensiuni mari - bloburi de date nestructurate cum ar fi muzică, filme, imagini grafice

DBMS_LOB : pachet - lucrul cu aceste tipuri

SELECT returnează date pt tipurile LONG
LONG RAW

SELECT returnează adresa (locatorul) pt LOB-uri

Ref cursor și ref obiect reprezintă adresa, locația de memorie a unui element de tip cursor sau obiect în sensul OOP.

Tipul obiect - atribute, metode

Tip nespecific

alias de legătură - se declară în mediul gazdă pt transferul valorilor în respectiv din unul sau mai multe programe PL/SQL

Exemplu:
```
VARIABLE x
BEGIN
  SELECT COUNT(*) INTO :x
  FROM ...
  WHERE ...
END;
PRINT x
```
(:x - alias)

Variabila gazdă - transferul valorilor între mediul de programare (precompilatorul) și comenzile SQL ce comunică cu serverul

Variabile indicator - permite comunicarea valorii NULL între programul scris în lb gazdă și sistemul ORACLE

Declararea variabilelor

- atributul %TYPE - ca cu celelalte tipuri ca o coloană de tabel
  %ROWTYPE ---u--- ca o linie de tabel

Utilizatorul poate să-și definească propriile tipuri și subtipuri în partea declarativă a unui bloc PL/SQL, subprogram sau pachet.

x := null;
Ce tip are x? Răspuns: Orice tip, deoarece null apare în toate tipurile.

Tipuri compuse - înregistrări RECORD
- pot fi asignate valori unui înregistrări utilizând SELECT / FETCH (cursoare) / :=
  o înregistrare poate fi atribuită altei înregistrări

Se poate (integra) o linie într-un tabel utilizând un record
Se poate reactualiza o linie a unui tabel utilizând record (sintaxa SET ROW)
Într-o înregistrare se poate → reține și returna
sau
→ șterge informația din clauza returning a comenzilor update și delete.

Colecții.
- tip tablou indexat (index by table) 1)
  tabel imbricat (nested table) 2)
  vector (varray) 3)

1) poate fi utilizat numai în declarații PL/SQL
2) 3) pot fi utilizate atât în declarații PL/SQL cât și în declarații la nivelul schemei (ca tip a unei coloane a unui tabel relațional)

Tabloul indexat în PL/SQL are 2 coloane: o col. cu cheie primară și o coloană cu valori efective

Tablou indexat → tabel relational
prin INSERT LOOP

tabel relational → tablou indexat
a) prin FETCH (cursoare)
b) liste de atribuiri (buclă)

* cum pot șterge liniile unui tablou indexat?
→ prin metoda delete
→ se asignează null componentelor sale ?!?
se declară un alt tablou indexat, neinițializat și acesta se asignează tabloului de șters

```
set serveroutput on
DECLARE
  TYPE tablou_numar IS TABLE OF NUMBER
       INDEX BY PLS_INTEGER;
  BEGIN
    FOR i IN 1..20 LOOP
        v_tablou(i):=i*i;
        DBMS_OUTPUT.PUT_LINE(v_tablou(i));
    END LOOP;
    FOR i IN v_tablou.FIRST..v_tablou.LAST LOOP
        v_tablou(i):=NULL;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('tabloul are', || v_tablou.COUNT    etc
```

<u>Vectori</u> - spre deosebire de tablouri indexate, au dim max stabilită la declarare care nu poate fi depășită
= depuși în memorie în afara BD
= sunt structuri dense, nu pot șterge elemente individual
(cum facem? indicii ∈ [1 .. lim_max]

```
DECLARE
  TYPE arcoiris IS VARRAY(5) OF VARCHAR2(10);
  N_arc arcoiris := arcoiris('alb', 'negro', 'roto', 'verde');

BEGIN
    V_arc(3) := 'roto';
    V_arc.EXTEND; -- agrega un elem null
    V_arc(5) := 'albaricoque';
    -- extiende los elem a la cant exact
    V_arc.EXTEND;
END;
/
```