

PL/SQL - este un limbaj de programare care:

1. asigură accesarea informației dintr-o bază relațională DB.
2. permite gruparea comenzielor într-un bloc unei de tratare a datelor

Problematice abordate

1. Concepte generale
2. Controlul execuției unui program
3. Tipuri de date
4. Cursori
5. Subprograme
6. Pachete / biblioteci
7. SQL dintr-unic
8. Declarații
9. Gestirea și tratarea erorilor.

Concepte generale

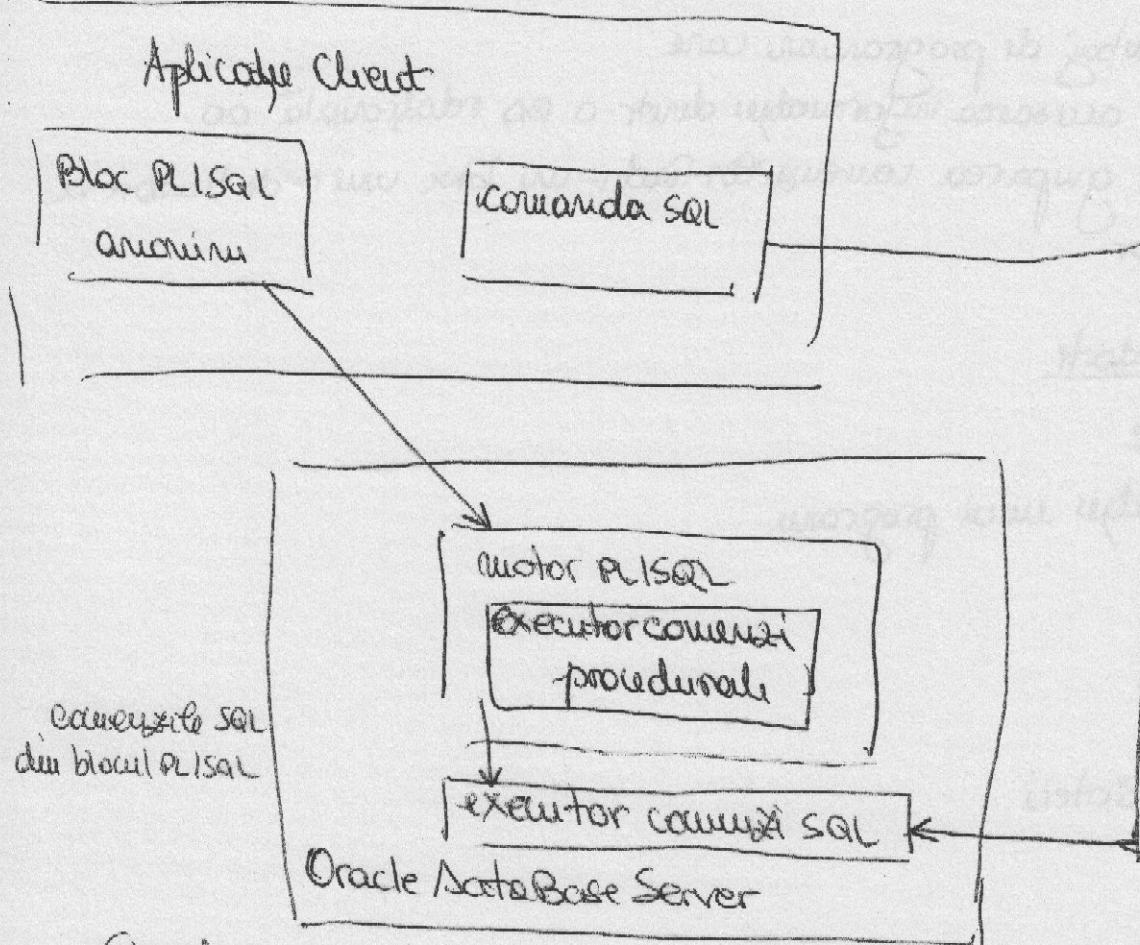
Blocurile PL/SQL sunt transmise unui motor PL/SQL * procesator (compilate, executate) de acesta. Motorul PL/SQL poate să se execute pe serverul Oracle sau îndr-un utilizator iar utilizarea se desfășoară de la fel ca și învocă. Blocurile PL/SQL pot fi executate pe statii client fără interacțiune cu serverul sau în întregime pe server. O aplicație BD poate fi structurată în

- interfață utilizator
- aplicație logică efectivă
- BD

Există 2 modele pentru proiectarea unei aplicații BD

- modelul client-server
- modelul three-tier

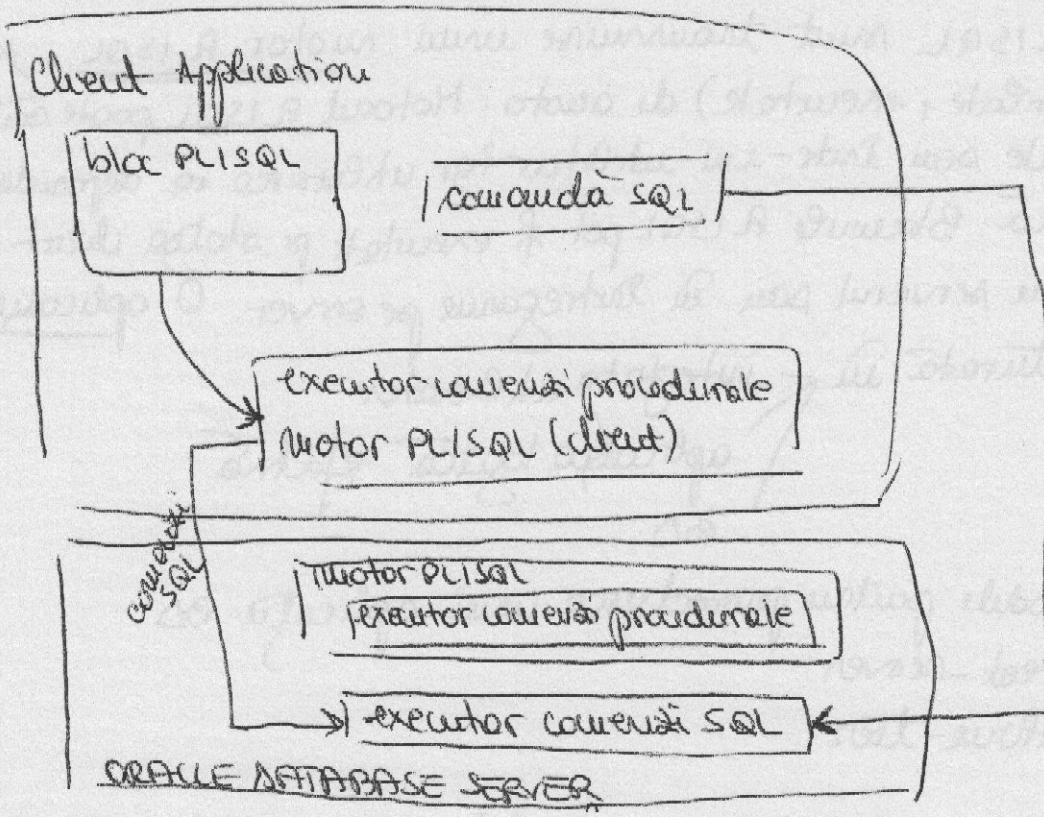
Motor PL/SQL server side



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



Datele aplicatiei sunt subprograme stocate, atunci este utilizat motorul PL/SQL de pe server

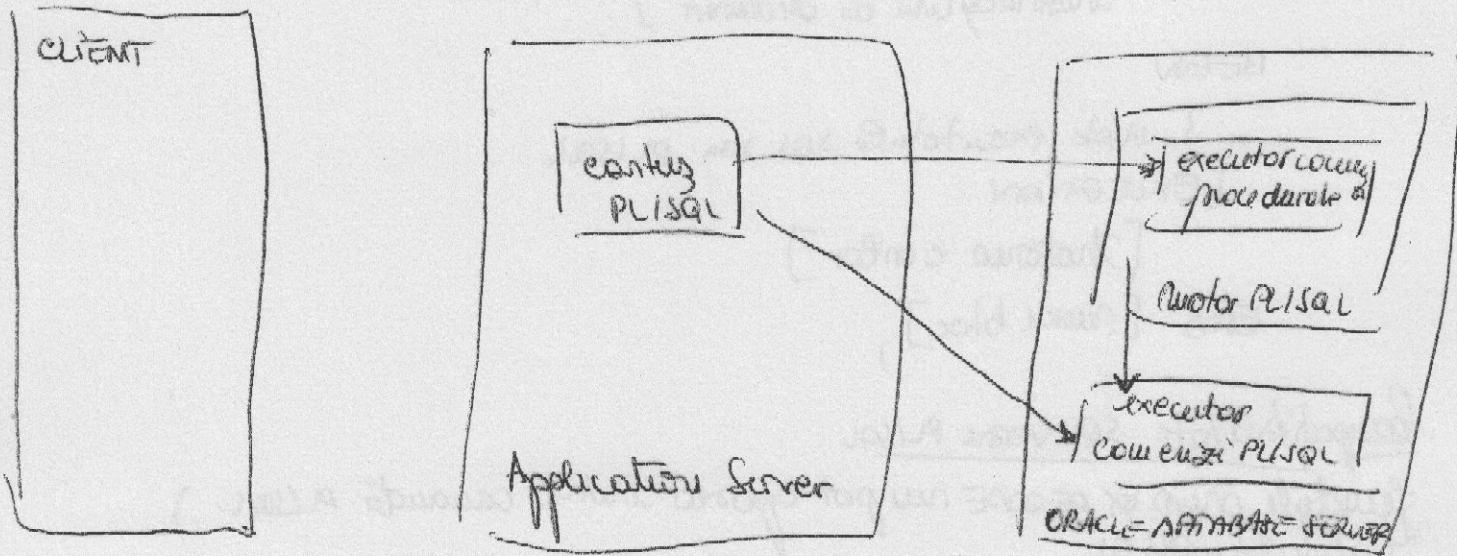
- consideram un scenario in care faza motoare PL/SQL: client local pe stocare client & client pe server

Ex: un trigger care se executa pe stocare client & care apela o procedura stocata in BD.

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

- aplicatia logica & BD sunt ximilate in 3 parti

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



functionul acronimelor ca pe client pt server

comenzile SQL & blocurile PL/SQL nu pot fi trimise serverului pt a fi executate. Pt a realiza acest lucru, tb. stabilite conexiunea la BD. PL/SQL nu permite nicio sintaxa care sa permita aceasta executie.

Pt ce fel de aplicatii este necesar PL/SQL?

- 1) proceduri & functii stocate
- 2) pachete
- 3) declaratorii BD
- 4) Application Trigger

Blocuri de executare unui program

Un program PL/SQL poate conține unul sau mai multe blocuri. Un bloc poate fi:

- reîncărcabile, blocuri etichetate, construite static sau dinamic și executate o singură dată
- subprograme
- pachete
- declaratorii

Structura bloc SQL

[*(nume bloc)*]

[DECLARARE

 instrucțiuni de declarație]

BEGIN

 } instrucțiuni executable SQL sau PL/SQL

[EXCEPTION

 [tratarea erorilor]

END [*(nume bloc)*].

Incompatibilitate SQL versus PL/SQL

Funcțiile și procedure nu pot opera într-o comandă PL/SQL

Incompatibilitate SQL

1. comunitate PL/SQL

2. opere

3. varialele globale

4. comunitate PL/SQL + {PL/SQL}

5. Blocul PL/SQL este o unitate transacțională? NU

6. PL/SQL nu suportă comunitate SQL primă ca și acordă/revoce
privilegii GRANT; REVOKE

7. SELECT *nu* returnează niciun rezultat NO DATA - Found

 | Autotext mai multe linii

 TOO_MANY_ROWS

 | Introduce o rezolvare

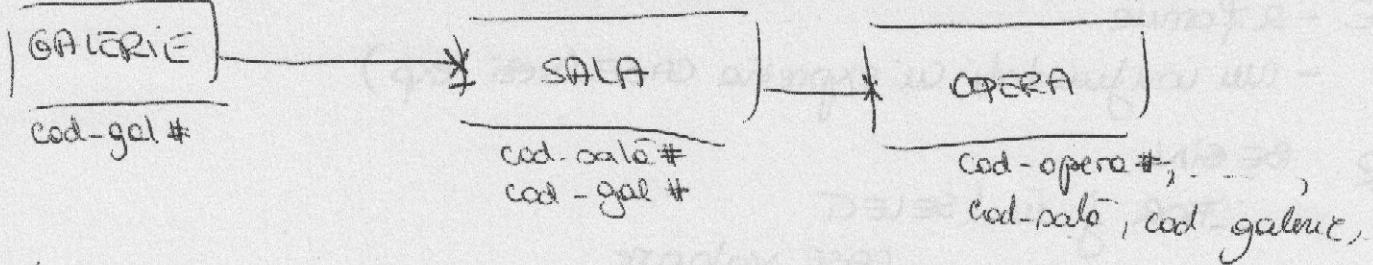
Clauza into în comandă select

SELECT

INTO *d1, d2, ...*

FROM

⑧ *ist* (variabila de lectură)



Băt se specifică dacă o galerie este mare medie sau mică depășind cu un operetor de ordine $\in [200, \text{întru } 100 - 200, < 100]$.

```

SET SERVEROUTPUT ON      → permite vizualizarea parțialului
DEFINIE p.cod_gal = 753
ABMS_OUTPUT
DECLARE
    v.cod_galerie opera.cod_galerie%TYPE := numar_label.varname numar_label.varname %TYPE;
    : = & p.cod_gal
    continut
    v.numar NUMBER(3) := 0;
    v.comentariu VARCHAR2(10);      aci 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 := 'medie';
    ELSE
        v.comentariu := 'mare';
    END IF;
    ABMS_OUTPUT.PUT_LINE ('Galeria având codul ' || v.cod_galerie ||
                           ' își face parte din ' || v.comentariu);
END;
    / → Încearcă și execută
    SET SERVEROUTPUT OFF
  
```

CASE - 2 forme

- cu valoare constantă sau expresie CASE (vizi 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) Instructionii Iterative loop → este obligatoriu să ducă la rezolvarea buclei
while
for

Loop

se execută în stările

END LOOP;

În structura de bază a operelor se va introduce un nou câmp (stea) să se adauge în bloc PL/SQL care va introduce o * pt fiecare 10000 \$ din valuri opera al cărui cod e specificat

ALTER TABLE opera

ADD stea VARCHAR2(20);

DEFINE p.cod-opere = 7777

DECLARE

v.cod-opere opera.cod-opere%TYPE := &p.cod-opere;

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-opere = v.cod-opere;

```

if v-valore>0 then
    for i in 1..v_valore LOOP
        v_stec := v_stec || '*';
    END LOOP;
END IF;
UPDATE opera
SET stec = v_stec
WHERE cod_opera = v_cod_opera ;
COMMIT;
END;

```

Instrucție salt : EXIT → reîncearcă dintr-un ciclu

EXIT [numele-etichetă] [WHEN conditie];

Instrucție NULL

```

IF (a = b) THEN
    NULL;
ELSE
    APPEND-OUTPUT.PUT-LINE('a > b');
END IF;

```

Instrucție GOTO