***PL/SQL – (PL-limbaj de programare)***

Limbajul SQL a devenit limbajul standard pentru lucrul cu baze de date deoarece este flexibil, puternic şi uşor de învățat. Datele stocate pot fi manipulate cu uşurinţă folosind câteva comenzi construite pe sintaxa limbii engleze, precum SELECT, INSERT, UPDATE, DELETE etc.

PL/SQL reprezintă extensia procedurală a SQL, şi a fost introdus de Oracle prima dată în 1991, odată cu SGBD-ul Oracle 6.0, evoluând continuu cu apariţia versiunilor ulterioare. Un limbaj procedural semnifică un limbaj de programare care permite controlul fluxului de execuţie prin instrucţiuni decizionale precum IF, cicluri (FOR), permite să declarăm variabile, să definim funcţii şi proceduri, să prelucrăm erorile de execuţie etc. PL/SQL permite folosirea tuturor comenzilor de manipulare a datelor din SQL şi utilizarea aceloraşi tipuri de date (NUMBER, VARCHAR2 etc.).

PL/SQL este un limbaj cu structura de bloc, adică programele pot fi împărţite în blocuri logice.

Un **bloc PL/SQL** este compus din trei secţiuni:

* **secţiunea declarativă** (opţională) conţine toate variabilele, constantele, cursoarele etc. la

care se face referinţă în secţiunea executabilă sau chiar în secţiunea declarativă;

* **secţiunea executabilă** conţine instrucţiuni SQL pentru manipularea datelor din baza de date şi instrucţiuni PL/SQL pentru manipularea variabilelor, constantelor etc.;
* **secţiunea pentru tratarea erorilor** (opţională) specifică acţiunile ce vor fi efectuate când în secţiunea executabilă apar erori.

Un program PL/SQL trebuie să conţină cel puţin un bloc. Acesta are următoarea structură generală:

**[ nume\_bloc ]**

**[ DECLARE**

**instrucţiuni de declarare ]**

**BEGIN**

**instrucţiuni executabile (SQL sau PL/SQL)**

**[ EXCEPTION**

**tratarea erorilor ]**

**END [ nume\_bloc ];**

Blocurile pot fi: anonime (fără nume), neanonime, proceduri, funcţii, triggere etc. De asemenea, un bloc poate conţine mai multe sub-blocuri.

Comentarii: -- sau /\* …\*/

Separator intre instructiuni: ;

Atribuire: :=

Operatori: +, -, \*,

Impartire reala: /

Ridicare la putere: \*\*

Concatenare: || concatenare

Comparare: =, !=, <, >, <=, >=, IS NULL, LIKE, BETWEEN, IN

SI/SAU logic: AND, OR

*Tipuri de date PL/SQL:*

* number(p,s), numeric(p,s), dec(p,s), decimal(p,s),
* float, double precision, real,
* int, integer, smallint,
* binary\_integer, natural, naturaln care exclude valoarea null,
* positive adica intreg >0,
* positiven (>0 si diferit de null),
* char, char(n), varchar2(n) unde n<4000 cu sinonimele string si varchar,
* Boolean =true, false, null
* Date,
* Record

*Sintaxa:*

**TYPE nume\_tip IS RECORD (camp1, camp2, …)**

**campk: nume\_tip [ [NOT NULL] {:= | Default} expresie]**

*Exemplu:*

**Declare**

**Type** angajat **is record(cod integer not null, nume varchar2 not null)**;

*Exemplu:*

**Declare**

**Type** angajat **is record(**cod integer, nume varchar2(40)**);**

a1 angajat;

**begin**

a1.cod:=101;

a1.nume:='Dan';

dbms\_output.put\_line(a1.cod || a1.nume);

**end**;

*Exemplu:*

***declare***

*r produse%rowtype;* /\*se creaza un record de tip produse \*/

***begin***

*r.cod:=7;*

*r.denumire:='feri';*

*r.pret:=15;*

*insert into produse values r;*

*select cod,denumire,pret into r from produse where cod=r.cod;*

*dbms\_output.put\_line(r.cod||' '||r.denumire||' '||r.pret);*

***end****;*

*Pachetul DBMS\_OUTPUT*

Este folosit pentru afisarea informatiilor (uzual pe ecran) când se execută blocuri și subprograme PL/SQL, ajutând la testarea și depanarea blocului.

Pachetul dbms\_output lucrează cu un buffer în care se scrie informația utilizând procedurile PUT, PUT\_LINE și NEW\_LINE.

Pachetul dbms\_output lucreaza cu variabila de sistem ***serveroutput*** care trebuie setata pe ***on*** pentru activarea afisarii pe ecran, utilizand sintaxa*:*

set serveroutput on [size N|unlimited];

exemple:

set serveroutput on size 30000;

set serveroutput on size unlimited;

*Sintaxa de declarare a variabilelor:*

**declare**

<identificator> [constant]

{tip\_date | identificator\_coloana%type | identificator\_tabel%rowtype}

[not null] [{:= | default expresie }

Exemplu:

**declare**

a date;

b number(5) not null :=100;

c varchar2(10):=’SGBD’;

pi constant number(8,7):=3.1415921;

d date default sysdate;

autor carte.autor%type; /\* se preia tipul din tabela carte

coloana autor \*/

rand\_carte carte%rowtype; /\*se creaza un record care are ca

tip tipul inregistrarii din tabela carte \*/

vocale constant char(5):=’aeiou’;

*Blocuri anonime:*

**Declare**

n natural;

c char(20);

**Begin**

c:='ec';

select count(marca) into n from Angajati where cod\_functie=c;

dbms\_output.put\_line(n);

/\*dbms\_output.put\_line pentru afisarea informatiilor in cadrul

blocurilor \*/

**end**;

***--instructiunea if***

IF conditie1 THEN

instructiuni1;

[ELSIF conditie2 THEN

instructiuni2;]

. . . . . . . . . . . . . . . . . . . .

[ELSE

instructiuni\_else;]

END IF;

**declare**

a int:=10;

b int:=20;

**begin**

if a>b

then dbms\_output.put\_line('a>b');

else dbms\_output.put\_line('b>a');

end if;

**end;**

*Rezolvarea ecuației de gradul 1: ax+b=0*

DECLARE

a NUMBER := 5;

b NUMBER := 7;

x NUMBER;

BEGIN

IF (a<>0) THEN

x := -b/a;

DBMS\_OUTPUT.PUT\_LINE(' x = ' || TO\_CHAR(x));

ELSE

IF (b<>0) THEN

DBMS\_OUTPUT.PUT\_LINE('Fara solutie');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Nedeterminare');

END IF;

END IF;

END;

**Instrucțiunea SELECT în PL/SQL**

Sintaxa instrucțiunii SELECT în PL/SQL este:

**SELECT lista\_coloane INTO *lista\_variabile***

**FROM lista\_tabele**

**[WHERE condiție]**

**[alte clauze]**

Clauza INTO precizează lista variabilelor care vor primi valori după executarea instrucţiunii.

Nu se pot prelua decât valori dintr-o singură linie rezultat la un moment dat (putem folosi SELECT…. INTO doar dacă instrucțiunea returnează o singură linie).

Numărul de variabile din *lista\_variabile* trebuie să fie egal cu numărul de coloane care apar în *lista\_coloane*.

*Exemplu*:

SET SERVEROUTPUT ON

DECLARE

Prenume VARCHAR2(20);

Nume VARCHAR2(10);

BEGIN

SELECT first\_name, last\_name INTO Prenume, Nume

FROM Employees

WHERE Employee\_ID = 200;

DBMS\_OUTPUT.PUT\_LINE('Nume si prenume: ' || Nume || ' ' || Prenume);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista date');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Mai multe linii');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Alta exceptie');

END;

Putem afişa numele unei erori (dacă nu cunoaştem identificatorul erorii ca în exemplul de mai sus) utilizând variabila Oracle *SQLERRM*:

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE(SQLERRM);

**declare**

departament angajati.cod\_dep%type := 'aprov';

total number(5);

total2 number(5);

**begin**

select count(marca) into total

from Angajati A where cod\_dep=departament;

dbms\_output.put\_line(departament || ' total ' || total ) ;

departament:= 'desf';

select count(marca) into total2

from Angajati A where cod\_dep=departament;

dbms\_output.put\_line(departament || ' total2 ' || total2 );

if total>total2

then dbms\_output.put\_line('aprovizionare');

else dbms\_output.put\_line('desfacere');

end if;

**end;**

***Instrucţiunea CASE:***

CASE expresie

WHEN 'val1' THEN Instructiuni1;

WHEN 'val2' THEN Instructiuni2;

……………………….

[ELSE

Instructiuni\_else;]

END CASE;

*Exemplu*:

SET SERVEROUTPUT ON;

DECLARE

Nota NUMBER(2);

BEGIN

Nota := 8;

CASE Nota

WHEN 10 THEN DBMS\_OUTPUT.PUT\_LINE('Excelent');

WHEN 9 THEN DBMS\_OUTPUT.PUT\_LINE('Foarte bine');

WHEN 8 THEN DBMS\_OUTPUT.PUT\_LINE('Bine');

WHEN 7 THEN DBMS\_OUTPUT.PUT\_LINE('Satisfacator');

WHEN 6 THEN DBMS\_OUTPUT.PUT\_LINE('Slab');

WHEN 5 THEN DBMS\_OUTPUT.PUT\_LINE('De trecere');

ELSE DBMS\_OUTPUT.PUT\_LINE('Nota prea mica (<5) sau Inexistenta (>10)');

END CASE;

END;

***Instrucțiuni repetitive***

***Instructiunea loop***

*Instrucţiunea LOOP are următoarea sintaxă:*

LOOP

Instructiuni;

EXIT [WHEN conditie]; -- ieșire din buclă când condiţie = TRUE

Instructiuni;

END LOOP;

declare

S integer;

i integer;

begin

S:=0;

i:=1;

loop

S:=S+i;

exit when S>100;

i:=i+1;

end loop;

dbms\_output.put\_line('S='||S||' '||'i='||i);

end;

***instructiunea while***

WHILE conditie

LOOP

Instructiuni;

END LOOP;

declare

S integer;

i integer;

begin

S:=0;

i:=0;

while S<100

loop

i:=i+1;

S:=S+i;

end loop;

dbms\_output.put\_line('S='||S||' '||'i='||i);

end;

***Instructiunea for***

*Sintaxa instrucţiunii FOR:*

FOR contor IN [REVERSE] valoare\_initiala .. valoare\_finala

LOOP

Instructiuni;

END LOOP;

declare

S integer;

i integer;

begin

S:=0;

for i in 1..100

loop

S:=S+i;

end loop;

dbms\_output.put\_line('S='||S||' '||'i='||i);

end;

**declare**

S integer;

i integer;

**begin**

S:=0;

for i in reverse 1..100

loop

S:=S+i;

end loop;

dbms\_output.put\_line('S='||S||' '||'i='||i);

**end**;

***Instructiunea goto***

declare

S integer;

i integer;

begin

S:=0;

i:=0;

<<start1>>

i:=i+1;

S:=S+i;

if S>100 then goto stop;

end if;

goto start1;

<<stop>>

dbms\_output.put\_line('S='||S||' '||'i='||i);

end;

**Funcţii uzuale PL/SQL**

***Funcţii ce operează pe şiruri de caractere***

**ASCII(caracter)** Returnează codul ascii al lui **caracter**

**CHR(cod\_ascii)** Returnează caracterul având codul ASCII furnizat

**CONCAT(sir1, sir2)** Concatenează cele două şiruri argumente

**LENGTH(sir)** Returnează lungimea şirului

**LPAD(sir,lungime\_totala,[caract])** Completează **sir** la stânga cu **caracter**, lungimea maximă a şirului astfel obţinut fiind **lung\_totala**. Dacă este omis parametrul **caracter** se consideră valoarea implicită spaţiu.

**RPAD(sir,lungime\_totala,[caract])** Comportare similară funcţiei **LPAD**, completarea realizăndu-se la dreapta argumentului **sir**.

***Funcţii numerice***

**ABS(numar)** Returnează valoarea absolută

**CEIL(numar)** Returnează cel mai mic număr întreg mai mare sau egal cu **numar**

**FLOOR(numar)** Returnează cel mai mare număr întreg mai mic sau egal cu **numar**

**ROUND(numar, [nr\_zec])** Rotunjeşte **numar** la numărul de zecimale specificat. Dacă **nr\_zec** este omis se va rotunji la întregul cel mai apropiat; dacă **nr\_zec**>0 rotunjirea se face la dreapta separatorului zecimal;

dacă **nr\_zec**<0 se va rotunji la stânga separatorului zecimal

**TRUNC(numar, [nr\_zec])** Truncheaza**numar** la numărul de zecimale specificat. Dacă **nr\_zec** este omis se va renunta la cifrele de dupa virgula; dacă **nr\_zec**>0 truncherea se face la dreapta separatorului zecimal;

dacă **nr\_zec**<0 truncherea se va face la stânga separatorului zecimal

***Funcţii de conversie***

**TO\_CHAR(data,format,[param\_NLS])** Converteşte o valoare **DATE** într-un şir **VARCHAR2**, permiţând prezentarea datei într-un anumit format. **param\_NLS** indică specificarea limbii utilizată la afişarea pe ecran. O parte din măştile de format ce pot fi utilizate sunt descrise mai jos.

**TO\_CHAR(num,format,[param\_NLS])** Converteşte orice număr în şir **VARCHAR2**. Formatele ce pot fi utilizate sunt descrise mai jos.

**TO\_DATE(sir,format,[NLS\_LANG])** Converteşte un şir de caractere (**CHAR** sau **VARCHAR2**), delimitat de apostrofuri, la o valoare **DATE**. Parametrul opţional **NLS\_LANG** permite precizarea limbii în care se va face afişarea.

Limitări:

- şirul argument nu poate avea mai mult de 220 caractere;

- suntem limitaţi la măştile din tabelul următor;

- nu putem combina formatele, cum ar fi specificarea formatului de 24 ore şi, în acelaşi timp, AM sau PM;

- nu putem specifica acelaşi element de două ori în şablonul de conversie (de exemplu, YYYY-MM-MMM-DD)

**TO\_NUMBER(sir,format,[param\_NLS])** Converteşte un şir de caractere (CHAR sau VARCHAR2) sau dată calendaristică într-un număr, cu scopul de a executa calcule numerice.Măştile de formatare sunt decrise în tabelul următor.

Şabloanele utilizate cu funcţiile **TO\_DATE**, **TO\_CHAR** şi**TO\_NUMBER**

***Măşti de formatare DATE***

**BC sau B.C.** Indicator Before Christ

**AD sau A.D.** Indicator Anno Domini

**SYYYY, YYYY** Anul cu 4 cifre. Întoarce un număr negativ dacă se utilizează **BC** cu **SYYYY**.

**YYY, YY, Y** Ultimele 3, 2 sau ultima cifră a reprezentării anului

**YEAR, SYEAR** Returnează anul, în litere. **SYEAR** returnează o valoare negativă dacă se utilizează cu **BC**

**MM** Numărul lunii, de la 01 la 12

**MONTH** Numele lunii, totdeauna pe 9 caractere, eventual completate la dreapta cu spaţii

**RM** Număr roman reprezentând luna, de la I la XII

**WW** Numărul săptămânii în an, de la 1 la 53

**W** Numărul săptămânii înlună, de la 1 la 5. Prima săptămână începe în prima zi a lunii

**D** Numărul zilei din săptămână, cu valori de la 1 la 7

**DD** Numărul zilei din lună, cu valori de la 1 la 31

**DDD** Numărul zilei din an, cu valori de la 1 la 366

**DAY** Numărul zilei din săptămână, în litere, totdeauna reprezentat pe 9 caractere, eventual completat la dreapta cu spaţii

**HH, HH12** Ora din zi, cu valori de la 1 la 12

**HH24** Ora din zi, cu valori de la 1 la 23

**MI** Numărul minutului din oră, de la 0 la 59

**SS** Numărul secundei din minut, de la 0 la 59

**AM** sau **A.M.** Indicator al orei ante-meridian

**PM** sau **P.M.** Indicator al orei post-meridian

***Măşti de formatare NUMERE***

|  |  |  |
| --- | --- | --- |
| **Mască** | **Exemplu** | **Descriere** |
| **9** | **9999** | Fiecare 9 este considerat o cifră semnificativă. Zerourile din faţa numărului sunt tratate drept spaţii |
| **0** | **09999** sau **99990** | Zerourile din stânga sau dreapta numărului sunt tratate şi afişate ca zerouri |
| **$** | **$999** | Indică poziţia punctului zecimal. Cifrele de 9 indică numărul maxim de cifre de o parte şi de alta a separatorului zecimal |

**Lucru cu date calendaristice:**

***select to\_char(sysdate, ’dd month yyyy’) as azi from dual;***

***select to\_char(sysdate, ’hh24:mi:ss’) as acum from dual;***

***sau***

***select to\_char(sysdate, ’hh:mi:ss’) as acum from dual;***

***select to\_char(sysdate-1, ’dd month yyyy’) as ieri from dual;***

***select to\_char(sysdate+1, ’dd mm yyyy’) as maine from dual;***

***select to\_char(to\_date(‘2014-12-15’, ’ yyyy mm dd ’ ), ‘day dd’) as data\_verificare from dual;***

*--convertire de la sir de char la date*

***select extract(year from sysdate) as anul from dual;***

***--cati angajati noi au fost in anul 2020/ in luna martie 2020??***

***select extract(month from sysdate) as luna from dual;***

***select extract(day from sysdate) as ziua from dual;***

***select to\_date(concat(’21/12’ ,extract(year from sysdate)),’dd/mm/yyyy’) as vacanta from dual;***

***--de inserat o data calendaristica intr-o coloana cu tipul date????***

***--afisare valori dintr-o coloana de tipul date sub un anumit format: 12-12-2020, 12/12/20,.....***

***select sysdate +5 from dual;***

***select add\_months(sysdate,5\*12)+10 from dual;*** *--adunam 5 ani si 10 zile*

***select length(‘decembrie’) from dual;***

***select initcap(‘decemBrIE’) from dual;*** *--sa inceapa cu litera mare*

***select upper(‘decemBrIE’) from dual;*** *--transforma in litere mari*

***---select \* from employees where upper(first\_name)=’CRISTIAN’***

***select lower(‘decemBrIE’) from dual;*** *--transforma in litere mici*

***select instr (‘decemBrIE’,’B’) from dual;*** *--returneaza pozitia unde gaseste caracterul ‘B’(case sensitive)*

***select substr(‘decembrie’,4,3) from dual;****--afiseaza 3 caractere incepand cu pozitia a 4a;*

***select to\_number(‘12’) +10 from dual;--****transforma din string in number*

***Functii PL/SQL***

| **Number** | **Character** | **Conversion** | **Date** |
| --- | --- | --- | --- |
| ABS  ACOS  ASIN  ATAN  ATAN2  BITAND  CEIL  COS  COSH  EXP  FLOOR  LN  LOG  MOD  POWER  ROUND  SIGN  SIN  SINH  SQRT  TAN  TANH  TRUNC | ASCII  CHR  CONCAT  INITCAP  INSTR  INSTRB  LENGTH  LENGTHB  LOWER  LPAD  LTRIM  NLS\_INITCAP  NLS\_LOWER  NLSSORT  NLS\_UPPER  REPLACE  RPAD  RTRIM  SOUNDEX  SUBSTR  SUBSTRB  TRANSLATE  TRIM  UPPER | CHARTOROWID  CONVERT  HEXTORAW  RAWTOHEX  ROWIDTOCHAR  TO\_BLOB  TO\_CHAR  TO\_CLOB  TO\_DATE  TO\_MULTI\_BYTE  TO\_NCLOB  TO\_NUMBER  TO\_SINGLE\_BYTE | ADD\_MONTHS  CURRENT\_DATE  CURRENT\_TIMESTAMP  DBTIMEZONE  EXTRACT  FROM\_TZ  LAST\_DAY  LOCALTIMESTAMP  MONTHS\_BETWEEN  NEW\_TIME  NEXT\_DAY  NUMTODSINTERVAL  NUMTOYMINTERVAL  ROUND  SESSIONTIMEZONE  SYSDATE  SYSTIMESTAMP  TO\_DSINTERVAL  TO\_TIMESTAMP  TO\_TIMESTAMP\_LTZ  TO\_TIMESTAMP\_TZ  TO\_YMINTERVAL  TZ\_OFFSET  TRUNC |

Probleme propuse

1) Scrieţi un program PL/SQL care să calculeze salariul mediu al angajaţilor programatori

(IT\_PROG) din tabela Employees, apoi să-l compare cu salariul unui angajat (de exemplu

cu ID = 200). Programul să afişeze: Salariul angajatului cu numele … este mai mare / mai

mic decât salariul mediu.

2) Scrieţi un program PL/SQL în care introducând (prin atribuire) un cod angajat din tabela

Employees, să afişeze anotimpul în care s-a angajat respectiva persoană. Puteţi utiliza

funcţia: TO\_CHAR(Data\_ang,'MM') pentru a extrage luna ca un număr.

3) Creaţi (înaintea programului PL/SQL) o tabelă BONUS(v\_id NUMBER, v\_salariu

NUMBER). Apoi, realizaţi un program PL/SQL care să insereze în tabela BONUS ID-ul şi

salariul majorat cu 10% al angajaţilor din tabela Employees, pentru ID-urile:

101,102,…,110.

4) Realizaţi un program care să actualizeze tabela BONUS în felul următor: salariile mai mari

de 9000 să fie diminuate cu 25%, iar salariile mai mici decât 6000 să fie majorate cu 10%.