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, …)

camp_k: 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;
```

Instrucţiuni 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 realizându-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	ASCII	CHARTOROWID	ADD_MONTHS
ACOS	CHR	CONVERT	CURRENT_DATE
ASIN	CONCAT	HEXTORAW	CURRENT_TIMESTAMP
ATAN	INITCAP	RAWTOHEX	DBTIMEZONE
ATAN2	INSTR	ROWIDTOCHAR	EXTRACT
BITAND	INSTRB	TO_BLOB	FROM_TZ
CEIL	LENGTH	TO_CHAR	LAST_DAY
COS	LENGTHB	TO_CLOB	LOCALTIMESTAMP
COSH	LOWER	TO_DATE	MONTHS_BETWEEN
EXP	LPAD	TO_MULTI_BYTE	NEW_TIME
FLOOR	LTRIM	TO_NCLOB	NEXT_DAY
LN	NLS_INITCAP	TO_NUMBER	NUMTODSINTERVAL
LOG	NLS_LOWER	TO_SINGLE_BYTE	NUMTOYMINTERVAL
MOD	NLSSORT		ROUND
POWER	NLS_UPPER		SESSIONTIMEZONE
ROUND	REPLACE		SYSDATE
SIGN	RPAD		SYSTIMESTAMP
SIN	RTRIM		TO_DSINTERVAL
SINH	SOUNDEX		TO_TIMESTAMP
SQRT	SUBSTR		TO_TIMESTAMP_LTZ
TAN	SUBSTRB		TO_TIMESTAMP_TZ

Number	Character	Conversion	Date
TANH	TRANSLATE		TO_YMINTERVAL
TRUNC	TRIM		TZ_OFFSET
	UPPER		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%.