Laborator 6
Limbajul PL/SQL.
Blocuri PL/SQL. Functii PL/SQL. Exceptii.

PL/SQL este limbajul procedural dezvoltat de Oracle ca o extensie a limbajului standard SQL, oferind o modalitate de execuţie a unor operaţii logice procedurale asupra bazelor de date.

Blocuri PL/SQL

PL/SQL este considerat un limbaj *structurat în blocuri*. Un bloc PL/SQL este o unitate sintactică care poate conţine cod program, declaraţii de variabile, secţiuni program de tratare a erorilor, apeluri de proceduri/funcţii sau chiar alte blocuri PL/SQL. Sintaxa generală minimală a unui bloc PL/SQL este următoarea:

```
DECLARE
     declaraţii  variabile
BEGIN
     cod  program       --comentarii in PL/SQL
END
/
```

Caracterul ”/” de la sfârşitul blocului conduce la compilarea şi executarea acestuia.

În cadrul clientului *SQL*Plus* este necesară şi setarea: Comanda SET ServerOutput ON seteaza trimiterea spre ecranul clientului a mesajelor afişate de funcţiile *PUT_LINE* sau *PUT* ale pachetului predefinit -*built_in*- DBMS_OUTPUT.

Blocul următor de cod PL/SQL afiseaza pe ecran continutul unei variabile *x*, initializata in cadrul blocului.

```
SET SERVEROUTPUT ON;
DECLARE
x INTEGER;
BEGIN
x:=1000;
DBMS_OUTPUT.PUT_LINE(x);
END;
/
```

Functii PL/SQL

Sintaxa generală a unei funcţii PL/SQL este următoarea:

```
CREATE [OR REPLACE] FUNCTION
       nume_functie[(definitii_parametrii_apel)]
RETURN (tip_parametru_returnat)
AS/IS
      [secţiune opţională declaraţii variabile interne functie]
BEGIN
      corp funcţie   -- cod program
  [EXCEPTION
      secţiune opţională de tratare excepţii]
END;
/
```

Caracterul ,/' de la finalul functiei conduce la compilarea functiei. Pentru a vizualiza posibilele erori care apar la compilare, se poate utiliza comanda *show errors*;

Caracteristici:

- Functiile sunt obiecte PL/SQL stocabile in baza de date;
- Caracteristica principala a unei funcţii PL/SQL este aceea că returnează obligatoriu un rezultat, de un anume tip bine precizat;
- In cazul unei funcţii cu diverse ramificaţii, este obligatorie returnarea unei valori (conform principiului de baza al unei funcţii) indiferent de ramura pe care se face ieşirea din corpul funcţiei.
- In secţiunea RETURN precizând tipul parametrului de ieşire returnat, nu trebuie precizată dimensiunea acestuia, ci doar tipul lui;
- Comanda PL/SQL RETURN (ultima executată în cadrul corpului propriu-zis al funcţiei), returnează spre programul apelant al funcţiei parametrul de ieşire, şi în acelaşi timp, asigură terminarea funcţiei (ieşirea din funcţie). Acest ultim aspect face ca, orice altă bucată de cod existentă după o comandă RETURN, să nu se mai execute.
- La definirea parametrilor de apel ai unei functii nu trebuie precizata dimenisunea, ci doar tipul lor;
- Într-o funcţie (spre deosebire de o procedură stocată) nu se pot folosi comenzi SQL (DML sau DDL);
- Singurul contact al functiilor cu continutul tabelelor unei baze de date are loc prin intermediul comenzilor SELECT...INTO...;
- Funcţiile pot fi folosite ca parte a unei expresii;

Rularea unei functii:

- Rularea unei functii se realizeaza printr-o comanda:

```
SELECT functie1(param1, param2..) FROM DUAL;
```

unde *DUAL* este o tabelă specială (virtuală) existentă în orice baza de date Oracle.

-O funcţie poate fi utilizată ca parte a unei expresii:

```
SELECT * FROM tabela WHERE camp1<functie1;
```

sau

```
SELECT camp1+functie1 FROM tabela;
```

- Rularea unei functii dintr-un bloc PL/SQL:

```
DECLARE
x integer :=100;
BEGIN
dbms_output.put_line(functie1(x));
END;
/
```

Observaţie: Oracle stochează orice erori pe care le găseşte într-o tabelă a bazei de date numită *USER_ERRORS*. Dacă se doresc detalii privind, spre exemplu, ultimele erori apărute la compilarea unei funcţii, se poate utiliza fie comanda **SHOW ERRORS**, fie se poate interoga tabela *USER_ERRORS* (exemplificare: SELECT LINE,TYPE, NAME, TEXT FROM USER_ERRORS;).

Despre exceptii:

- Sectiunea EXCEPTION este dedicata tratarii erorilor returnate de serverul Oracle sau a diferitelor exceptii definite de utilizator;
- Sectiunea EXCEPTION este o secvenţă de cod PL/SQL specifică atât funcţiilor, procedurilor stocate, trigger-elor cât şi practic oricărui bloc de cod PL/SQL.
- Secţiunea *EXCEPTION* este opţională;
- Exceptiile pot avea asociat un nume (identificator). Unele dintre aceste nume sunt predefinite, pentru o parte dintre erorile returnate de server. De exemplu exceptia ZERO_DIVIDE.
- Exceptia ZERO_DIVIDE este o exceptie predefinita care se declanseaza in momentul in care in codul obiectului PL/SQL se intalneste o impartire la zero.
- Exceptia NO_DATA_FOUND se declanseaza atunci cand cererea SELECT care ar fi trebuit sa returneze o linie, nu returneaza nicio linie;
- In cadrul codului unui obiect PL/SQL, este posibila declararea propriilor excepţii, iar în secţiunea EXCEPTION, cu o clauza WHEN acestea pot fi tratate separat;
-Apelul unei excepţii se face prin comanda: *RAISE nume_exceptie_anterior_declarata (e*vident, in acest caz excepţia este generata manual, prin cod, funcţie de o anumită condiţie logică, şi nu funcţie de încălcarea unor reguli de integritate a datelor);
- Exceptia OTHERS este o exceptie pe care baza de date o aruncă în cazul oricărei excepții din afara celor predefinite și definite de către user. Ramura WHEN OTHERS, intercepteaza orice eroare (excepţie) care nu este tratată distinct explicit. Aceasta ramura, daca exista, este intotdeauna ultima.
- Pe fiecare ramură corespunzătoare secţiunii de excepţie, dacă există, funcţiile PL/SQL trebuie să returneze ceva de tipul precizat.
-Aparitia unei exceptii este numita si ***ridicarea unei exceptii*** (exception raising);
-Aparitia unei exceptii, tratata sau nu, determina terminarea executiei blocului. Toate instructiunile PL/SQL sau comenzile SQL care apar dupa cererea / instructiunea care a ridicat exceptia nu mai sunt executate;
- In cazul in care o exceptie are asociat un tratament, dupa ridicarea ei se trece la executia instructiunilor de tratare, dupa care se iese din blocul curent fara eroare;
- In cazul in care apare o exceptie care nu este tratata explicit in portiunea dintre EXCEPTION si END, blocul se termina cu eroare;
- În cazul unei excepţii generate manual prin cod (RAISE_APPLICATION_ERROR), funcţie de îndeplinirea sau nu a unei condiţii logice, un ROLLBACK implicit este executat automat (fără a fi necesară precizarea lui explicită în cod).
- În cazul unei excepţii generate de o încălcare a unor constrângeri de integritate a datelor, comanda care a generat o astfel de excepţie va eşua (şi evident un ROLBACK implicit este executat).

Sinteza:

- Exceptiile sunt de trei feluri:
  - Exceptii Oracle predefinite;
  - Exceptii Oracle care nu sunt predefinite;
  - Exceptii definite de utilizator.

1. Exceptii Oracle predefinite:
- au deja un identificator asociat. Ele sunt ridicate automat la aparitie de catre serverul Oracle;

- NU se declara in zona DECLARE;
- NU trebuie ridicate explicit prin RAISE;
- Se trateaza sectiunea EXCEPTION;

2. Exceptii Oracle non-predefinite:

Reprezinta alte erori returnate de Oracle, in afara celor predefinite. Aceste erori se pot trata in doua moduri:
- Prin folosirea lui WHEN OTHERS
- Prin declararea exceptiei (atribuirea unui nume) si asocierea unui cod de eroare Oracle in zona DECLARE, urmata de tratarea ei in sectiunea EXCEPTION.

- Se pot declara in zona DECLARE si li se poate asocia un cod de eroare;
- NU trebuiesc ridicate explicit prin RAISE;
-Se trateaza in zona EXCEPTION;

3. Exceptii definite de utilizator:

Un program PL/SQL poate contine exceptii definite de utilizator. In cazul cand programul detecteaza o situatie anormala el poate trece in zona de exceptii prin instructiunea PL/SQL RAISE nume_exceptie. Astfel de exceptii trebuie declarate explicit si ridicate explicit.

- Se declara in zona DECLARE (ca nume);
- Se ridica explicit cu RAISE;
- Se trateaza in zona EXCEPTION;

Definirea unor erori ad-hoc

Erorile ad-hoc sunt similare erorilor Oracle non-predefinite dar trebuiesc ridicate explicit in zona executabila sau in zona de exceptii folosind:

RAISE_APPLICATION_ERROR(cod_eroare, text_eroare);

unde:

- cod_eroare este un numar ales de utilizator intre -20000 si -20999
- text_eroare este un text ales de utilizator

Probleme rezolvate:

1). Sa se creeze o functie PL/SQL, numita C*lasificare,* care va stabili daca valoarea primita ca si parametru de intrare (de tip integer) este un numar par sau impar. Functia va returna mesajul 'Numar par', daca numarul primit ca si parametru este par, respectiv 'Numar impar', daca numarul primit ca si parametru este impar. Pentru verificarea daca numarul este par sau impar, se va folosi functia *mod*.

Observatii:
- Tipul de data returnat de functie se va considera varchar;
- Se recomanda utilizarea sectiunii de tratare a exceptiilor EXCEPTION, astfel:
  - in cazul in care numarul nu este nici par, nici impar, se va declansa o exceptie definita de utilizator, numita, *er* (*er* - variabila de tip EXCEPTION). Tratarea acesteia se va face pe ramura WHEN *er* THEN, prin afisarea mesajului 'Parametru invalid!';
  - se cere utilizarea exceptiei non-predefinite *OTHERS,* interceptand orice eroare care nu este tratata distinct explicit.
- Rularea functiei se va face cu diversi parametri, pentru a se evidentia atat identificarea numerelor pare si impare, cat si situatiile de exceptie care pot aparea (parametru invalid, sau orice alte exceptii).

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE FUNCTION Clasificare (nr integer)
RETURN varchar IS
er EXCEPTION;
BEGIN
 If (nr mod 2)=0 THEN
    RETURN 'Numar par';
 ELSIF (nr mod 2)=1 THEN
    RETURN 'Numar impar';
 ELSE
    RAISE er;
 END IF;

EXCEPTION
WHEN er THEN
  RETURN 'Parametru invalid!';
WHEN OTHERS THEN
  RETURN 'Eroare!';
END;
/
```

```
SQL> select Clasificare(5) as Rezultat from dual;

REZULTAT
--------------------------------------------------------------------------------
Numar impar

SQL> select Clasificare(2) as Rezultat from dual;

REZULTAT
--------------------------------------------------------------------------------
Numar par
```

```
SQL> select Clasificare(2.5) as Rezultat from dual;

REZULTAT
--------------------------------------------------------------------------------
Parametru invalid!
```

2). Sa se creeze o functie PL/SQL, numita *operatii*, care primeste trei parametri: primul parametru (operatorul - de tip char) stabileste tipul operatiei care se efectueaza cu ceilalalti doi parametri (operanzii – de tip number). Functia poate realiza operatia de adunare, scadere, inmultire sau diviziune, returnand rezultatul operatiei.

Observatii:
- Tipul de data returnat de functie se va considera varchar, conversia de la tipul number (rezultatul operatiei aritmetice) la varchar, facandu-se automat;
- In cazul in care operatorul primit ca si parametru este diferit de simbolul '+', '-', '*', sau '/', functia va returna mesajul "Operator invalid";
- Se recomanda utilizarea sectiunii de tratare a exceptiilor EXCEPTION, astfel:
  - in cazul in care functia realizeaza operatia de diviziune, iar al doilea operand este 0, se doreste folosirea exceptiei predefinite *zero_divide;*
  - se cere utilizarea exceptiei non-predefinite *OTHERS,* interceptand orice eroare care nu este tratata distinct explicit.
- Rularea functiei se va face cu diversi parametri, pentru a se evidentia atat operatiile de inmultire/impartire/ adunare/scadere valide, cat si situatiile de exceptie care pot aparea (operator invalid, impartire la zero, sau orice alte exceptii).

```
CREATE OR REPLACE FUNCTION operatii (var0 char, var1 number, var2 number)
RETURN varchar IS
rezultat number;
BEGIN
 IF var0='*' THEN
    rezultat:=var1*var2;
    RETURN rezultat;
 ELSIF var0='/' THEN
    rezultat:=var1/var2;
    RETURN rezultat;
 ELSIF var0='-' THEN
    rezultat:=var1-var2;
    RETURN rezultat;
 ELSIF var0='+' THEN
    rezultat:=var1+var2;
    RETURN rezultat;
 ELSE
    RETURN 'Operator invalid!';
 END IF;


EXCEPTION
WHEN zero_divide THEN
  RETURN 'Impartire la zero!';
WHEN OTHERS THEN
  RETURN 'Eroare!';
END;
/
```

```
SQL> select operatii ('*',10,12.25) from dual;

OPERATII('*',10,12.25)
----------------------------------------
122.5

SQL> select operatii ('/',10,12.25) from dual;

OPERATII('/',10,12.25)
----------------------------------------
.816326530612244897959183673469387755102

SQL> select operatii ('/',10,0) from dual;

OPERATII('/',10,0)
----------------------------------------
Impartire la zero!

SQL> select operatii ('+',10,2) from dual;

OPERATII('+',10,2)
----------------------------------------
12

SQL> select operatii ('-',10,3) from dual;

OPERATII('-',10,3)
----------------------------------------
7

SQL> select operatii ('p',10,0) from dual;

OPERATII('p',10,0)
----------------------------------------
Operator invalid!
```

3). Se considera o tabela ORACLE, numita Clinica, avand campurile:
-CNP char(13) PRIMARY KEY,
-nume varchar (30) NOT NULL.

Tabela contine urmatoarele 2 inregistrari:

CNP	Nume
2830812445577	Laudatu Anca
2810517445577	Mot Razvan

```
SET AUTOCOMMIT ON;
SELECT SYSDATE FROM DUAL;
ALTER SESSION SET NLS_DATE_FORMAT='DD-MM'YY';

CREATE TABLE Clinica (
CNP CHAR(13) PRIMARY KEY, nume VARCHAR (30) NOT NULL);
```

```
INSERT INTO CLINICA VALUES('2830812445577' ,'Laudatu Anca');
INSERT INTO CLINICA VALUES('2810517445577' ,'Mot Razvan');
```

Sa se creeze o functie PL/SQL, numita Data_nasterii, care va primi ca si parametru de intrare CNP-ul pacientilor clinicii medicale si va returna data nasterii acestora.

```
CREATE OR REPLACE FUNCTION Data_nasterii (cnp varchar)
RETURN date IS
cnp_substring char(6);
data_nasterii date;

BEGIN

 cnp_substring:=substr(cnp,6,2)|| substr(cnp,4,2)|| substr(cnp,2,2);
 -- || - reprezinta operatorul de concatenare in PL/SQL
 data_nasterii:=to_date(cnp_substring,'dd-mm-yy');
 RETURN data_nasterii;

END;
/
```

Sa se ruleze functia cu parametru de apel '2830812445577':

```
SQL> select Data_nasterii ('2830812445577') as DATA from dual;

DATA
--------
12-08-83
```

Sa se apeleze functia Data_nasterii dintr-un bloc PL/SQL:

```
SET SERVEROUTPUT ON;
 DECLARE
 cnp_ext CHAR(13):='2830812445577';
 nume_p varchar(30);
 BEGIN
 SELECT nume INTO nume_p  FROM Clinica WHERE CNP=cnp_ext;
 DBMS_OUTPUT.PUT_LINE(nume_p||' '||cnp_ext||'  '|| Data_nasterii(cnp_ext));
 END;
  /

Rezultat:
Laudatu Anca 2830812445577  12-08-83
```

Sa se afiseze CNP-ul , numele si data nasterii pentru toti pacientii din tabela Clinica.

```
SELECT CNP, nume, Data_nasterii(CNP) as DATA from Clinica;

CNP                NUME                 DATA
……..……...
2830812445577   Laudatu Anca          12-08-83
2810517445577   Mot Razvan            17-05-81
```

Probleme propuse:

1. Se considera tabela Salarii, avand campurile:

-CNP char(13) PRIMARY KEY,
-Nume varchar (30) NOT NULL,
-Functie varchar(20) NOT NULL,
-Venit_brut numeric NOT NULL,
-Venit_net numeric.

Sa se introduca 2 inregistrari in tabela, dupa cum urmeaza:

```
SET AUTOCOMMIT ON;
CREATE TABLE Salarii (
CNP char(13) PRIMARY KEY,
Nume varchar(30) NOT NULL,
Functie varchar(20) NOT NULL,
Venit_brut numeric NOT NULL,
Venit_net numeric);

INSERT INTO Salarii VALUES('111', 'Vali', 'inginer',4400, NULL);
INSERT INTO Salarii VALUES('222', 'Alina', 'inginer',2200, NULL);
```

Sa se scrie o functie PL/SQL, numita Salariu_net, care primeste ca si parametru CNP-ul unui angajat din tabela Salarii, calculeaza venitul net al acestuia si afiseaza urmatoarele informatii sub forma:

SQL> SELECT Salariu_net ('111') FROM DUAL;

SALARIU_NET('111')

2574

CNP: 111
Nume: Vali
Functie: inginer
Venit brut: 4400
CAS: 1100
CASS: 440
Venit brut impozabil: 2860
Impozit: 286
Venit net: 2574

Venitul net se calculeaza ca fiind salariul brut impozabil – IMPOZIT. IMPOZIT-ul reprezinta 10% din salariul brut impozabil. Salariul brut impozabil se calculeaza ca fiind salariul brut – CAS-CASS. CAS reprezinta 25% din salariul brut, iar CASS reprezinta 10% din salariul brut.

Observatii:
- Tipul de data returnat de functie se va considera numeric;
- Se recomanda crearea sectiunii de tratare a exceptiilor EXCEPTION si tratarea exceptiei predefinite NO_DATA_FOUND care se declanseaza in situatia in care comenzile select utilizate in cadrul functiei nu returneaza nicio linie, respectiv tratarea exceptiei non-

predefinite *OTHERS,* interceptand orice eroare care nu este tratata distinct explicit. In cazul declansarii exceptiei predefinite NO_DATA_FOUND, pe ramura WHEN NO_DATA_FOUND, se cere afisarea mesajului ‘Nu exista astfel de date' si returnarea valorii 0. In cazul aparitiei oricarei alte exceptii (pe ramura WHEN OTHERS) se cere afisarea mesajului ,’A aparut o alta exceptie!’ si returnarea valorii 0.

2. Se considera tabela Produse, avand campurile:

-Id_produs integer PRIMARY KEY,
-Nume_produs varchar (30) NOT NULL,
-Pret numeric NOT NULL,
-Cantitate numeric NOT NULL CHECK (Cantitate >=0),

Sa se introduca 3 produse in tabela, dupa cum urmeaza:

```
SET AUTOCOMMIT ON;
CREATE TABLE Produse (
Id_produs integer PRIMARY KEY,
Nume_produs varchar(30) NOT NULL,
Pret numeric NOT NULL,
Cantitate integer NOT NULL CHECK (Cantitate >=0));

INSERT INTO Produse VALUES(1, ’Iaurt’, 15, 20);
INSERT INTO Produse VALUES(2, ’Lapte’, 20,4);
INSERT INTO Produse VALUES(3, ’Paine’, 10,0);
```

Sa se scrie o functie PL/SQL, numita Stoc, care primeste ca si parametru Id –ul unui produs din tabela Produse si verifica stocu-ul produsului respectiv. Daca stocul e mai mare sau egal cu 5 bucati, atunci functia va returna mesajul ‘Produs in stoc!’. Daca stocul e mai mic decat 5 bucati, dar mai mare decat 0, functia va returna mesajul ‘Stoc la limita (sub 5 bucati)!’. Daca stocul este egal cu 0, functia va afisa mesajul ‘Stoc epuizat!’.

Observatii:
- Tipul de data returnat de functie se va considera varchar;
- Se recomanda crearea sectiunii de tratare a exceptiilor EXCEPTION si tratarea exceptiei predefinite NO_DATA_FOUND care se declanseaza in situatia in care produsul cautat nu exista in tabela Produse, respectiv tratarea exceptiei non-predefinite *OTHERS,* interceptand orice eroare care nu este tratata distinct explicit. In cazul in care produsul cautat nu exista in tabela Produse (pe ramura WHEN NO_DATA_FOUND), functia va returna mesajul ‘Nu exista produsul cautat!’. In cazul in care apare o alta exceptie in timpul rularii functiei (pe ramura WHEN OTHERS), functia va returna mesajul 'A aparut o alta exceptie!’.

```
SQL> select stoc(1) from dual;

STOC(1)
--------------------------------------------------------------------------------
Produs in stoc!


SQL> select stoc(2) from dual;

STOC(2)
--------------------------------------------------------------------------------
```

```
Stoc la limita (sub 5 bucati)!

SQL> select stoc(3) from dual;

STOC(3)
--------------------------------------------------------------------------------------------------------------
Stoc epuizat!

SQL> select stoc(4) from dual;

STOC(4)
-----------------------------------------------------------------------------------------------
Nu exista produsul cautat!
```

Sa se afiseze toate produsele din tabela Produse, sub forma:

Id_produs	Nume_produs	Pret	Status
1	Iaurt	15	Produs in stoc!
2	Lapte	20	Stoc la limita (sub 5 bucati)!
3	Paine	10	Stoc epuizat!