Laborator 7
Limbajul PL/SQL. Proceduri stocate.

O procedura stocata este o succesiune/secvenţa de comenzi SQL-PL/SQL, stocata in baza de date, pe serverul SQL - intr-un obiect PL/SQL dedicat Oracle. Procedurile stocate executa un task bine definit, fiind apelate in mod explicit de către un client SQL sau dintr-o alta secvenţa de cod. O *procedură stocată* este compilata si salvata in baza de date, in momentul compilării, ea devenind un obiect din *schema user*-ului care a creat-o (obiect al bazei de date).

Sintaxa generală a unei proceduri stocate PL/SQL este următoarea:

```
CREATE [OR REPLACE] PROCEDURE
 nume_procedura[(definitii_parametrii_apel)]
AS/IS
[sectiune optională declaratii variabile interne]
BEGIN
      corp procedura
[EXCEPTION
      sectiune optională de tratare exceptii]
END;
/
```

Se pot remarca în cadrul definiţiei unei proceduri stocate două componente primare: secţiunea de specificare a numelui procedurii si, opţional, a parametrilor ei de apel, respectiv corpul procedurii, care conţine comenzi SQL şi/sau PL/SQL asigurând operaţiile dorite.

O procedură stocată nu poate fi modificată, ea trebuind fie să fie înlocuită printr-o nouă definiţie, fie trebuie eliminată şi apoi recreată (*DROP* şi *CREATE*). Când se înlocuieşte o procedura, trebuie inclusă clauza *OR REPLACE* în sintaxa comenzii *CREATE PROCEDURE*.

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

Execuţia unei proceduri stocate se poate face utilizand o comandă de genul:

```
EXEC[UTE] nume_procedura (lista_parametrii_intrare);
```

O procedură stocată poate fi apelată şi din cadrul corpului altei proceduri stocate În acest caz însă, apelul se face doar prin simplă precizare a numelui procedurii apelate (şi eventual a parametrilor de apel), fără a se mai utiliza cuvântul cheie *EXECUTE* sau forma scurtă *EXEC*.

Caracteristici:

- Procedurile stocate diferă de funcţiile SQL prin faptul că nu returnează valori (utilizabile în locul numelui lor), neputând fi utilizate direct într-o expresie. Există însă şi în cadrul unei proceduri parametrii de apel de tip OUT, similari celor de la funcţii.
- Comanda RETURN de la sfârşitul unei proceduri este implicită. În cadrul unei proceduri, RETURN nu poate conţine o expresie (lucru evident din moment ce o procedură nu returnează nimic).

- Intr-o procedură stocată nu poate fi utilizată o interogare SQL clasică *SELECT* (returnând date spre un client). Poate fi utilizată în schimb o comandă *SELECT...INTO*, returnând date spre una sau mai multe variabile PL/SQL.
- Intr-o procedura pot fi utilizate comenzile SQL DML (*INSERT, DELETE, UPDATE*).
- Sectiunea EXCEPTION, prezentata in detaliu in lucrarea 6 de laborator poate fi utilizata si in cadrul procedurilor stocate, ea fiind specifică atât funcţiilor, procedurilor stocate, trigger-elor cât şi practic oricărui bloc de cod PL/SQL.

Probleme rezolvate:

1). Scrieti o procedura PL/SQL, numita *afisare*, al carei apel sa determine afisarea pe ecran a parametrului sau de intrare (de tip varchar).

Observatii:
- Se cere folosirea constructiei DBMS_OUTPUT.PUT_LINE.
- Se cere apelarea procedurii atat direct, din client, cu ajutorul comenzii *exec*, cat si dintr-un bloc PL/SQL.

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE afisare (var varchar) AS
BEGIN
DBMS_OUTPUT.PUT_LINE(var);
END;
/
```

```
SQL> exec afisare('mesaj');
mesaj

SQL> exec afisare('1212');
1212

--apel procedura dintr-un bloc PL/SQL

 DECLARE
 var varchar(30):= 'Hello world!';
 BEGIN
 afisare(var);
 END;
 /

Hello world!
PL/SQL procedure successfully completed.
```

2). Considerand tabela *Evidenta_persoane*, cu doua campuri (*id* de tip integer, cheie primara, si *nume* de tip varchar(30)), scrieti o procedura stocata, numita *adaugare*, care permite adaugarea de noi linii in tabela. Se vor trata exceptiile care pot sa apara, abandonand operatia si afisand un mesaj de eroare.

```
CREATE TABLE Evidenta_persoane (id INTEGER PRIMARY KEY, nume VARCHAR(30));

CREATE OR REPLACE PROCEDURE adaugare (id integer, nume varchar) AS
BEGIN
INSERT INTO Evidenta_persoane VALUES(id,nume);
EXCEPTION
WHEN others THEN
   RAISE_APPLICATION_ERROR(-20000, 'Adaugare nereusita!');
END;
/
```

3). Considerind tabela *Evidenta_persoane*, cu doua campuri (*id* de tip integer, cheie primara, si *nume* de tip varchar(30)), scrieti o procedura stocata, numita *adaugare*, care permite adaugarea de noi linii in tabela. Se vor trata exceptiile care pot sa apara, permitand o adaugare in campul de tip primary key a valorii celei mai mari din el incrementata cu unu (ignorand valoarea pasata gresit ca parametru de intrare); se va semnala printr-un mesaj anomalia rezolvata.

```
CREATE TABLE Evidenta_persoane (id INTEGER PRIMARY KEY, nume VARCHAR(30));

CREATE OR REPLACE PROCEDURE adaugare (id integer, nume varchar) AS
var int;
BEGIN
INSERT INTO Evidenta_persoane VALUES(id,nume);
EXCEPTION
WHEN others THEN
BEGIN
  SELECT MAX(id) INTO var FROM Evidenta_persoane;
  INSERT INTO Evidenta_persoane VALUES(var+1,nume);
  DBMS_OUTPUT.PUT_LINE('Inserare reusita! ');
END;
END;
/
```

4). Fie tabela *pacienti*, avand campurile(cnp char(13) primary key, nume varchar (30), data_nasterii date). Creati o procedura stocata, numita *adaugare*, care verifica in momentul introducerii uni noi linii in tabela ca *data_nasterii* sa fie mai mica decat data curenta. Daca *data_nasterii*>data curenta, se completeaza campul cu NULL si se semnaleaza problema printr-un mesaj (linia adaugandu-se totusi).

```
CREATE TABLE pacienti (cnp char(13) PRIMARY KEY, nume VARCHAR(30),
data_nasterii date);

CREATE OR REPLACE PROCEDURE adaugare(cnp CHAR, nume VARCHAR, data_nasterii
DATE) IS
BEGIN
If data_nasterii>SYSDATE THEN
INSERT INTO pacienti VALUES(cnp,nume,NULL);
DBMS_OUTPUT.PUT_LINE('Inserare cu NULL!');
ELSE
INSERT INTO pacienti VALUES(cnp,nume,data_nasterii);
END IF;
EXCEPTION
WHEN others THEN
RAISE_APPLICATION_ERROR(-20000, 'Adaugare nereusita!');
END;
/
```

5). Fie tabela *pacienti*, avand campurile (cnp char(13) primary key, nume varchar (30), data_nasterii date). Creati o procedura stocata, numita *adaugare,* care verifica in momentul introducerii uni noi linii in tabela ca *data_nasterii* sa fie mai mica decat data curenta. Daca *data_nasterii*>data curenta, se interzice adaugarea noii linii, semnalandu-se acest lucru printr-un mesaj.

Observatii:
- Pentru rezolvarea situatiei in care data nasterii e mai mare decat data curenta, se cere generarea propriei exceptii;

```
CREATE OR REPLACE PROCEDURE adaugare(cnp CHAR, nume VARCHAR, data_nasterii
DATE) IS
er EXCEPTION;
BEGIN
If data_nasterii<SYSDATE THEN
  INSERT INTO pacienti VALUES(cnp,nume,data_nasterii);
ELSE
RAISE er;
END IF;
EXCEPTION
WHEN er THEN
  DBMS_OUTPUT.PUT_LINE('Adaugarea nu s-a efectuat. Data nasterii > Data
  curenta!');
WHEN others THEN
  RAISE_APPLICATION_ERROR(-20000, 'Eroare!');
END;
/
```

Probleme propuse:

1). Se considera baza de date XE, avand urmatoarea structura:

- Tabela studenti:
  - Marca char(6) PRIMARY KEY,
  - Nume varchar (30) NOT NULL,
  - An integer NOT NULL CHECK (An>0 and An<5),
  - Grupa integer NOT NULL,
  - Media_generala NUMERIC.

- Tabela discipline
  - Cod_disciplina integer PRIMARY KEY,
  - Denumire_disciplina varchar(30) NOT NULL.

- Tabela note
  - Marca char(6) NOT NULL REFERENCES studenti (Marca) ON DELETE CASCADE,
  - Nota integer NOT NULL CHECK (Nota >0 AND Nota <=10),
  - Cod_disciplina integer NOT NULL REFERENCES discipline(Cod_disciplina) ON DELETE CASCADE.

```
CREATE TABLE studenti (
Marca char(6) PRIMARY KEY,
Nume varchar(30) NOT NULL,
An integer NOT NULL CHECK (An>0 and An<5),
Grupa integer NOT NULL,
Media_generala NUMBER(4,2));

CREATE TABLE discipline (
Cod_disciplina integer PRIMARY KEY,
Denumire_disciplina varchar(30) NOT NULL);

CREATE TABLE note(
Marca char(6) NOT NULL REFERENCES studenti (Marca) ON DELETE CASCADE,
Nota integer NOT NULL CHECK (Nota >0 AND Nota <=10),
Cod_disciplina integer NOT NULL REFERENCES discipline(Cod_disciplina)
ON DELETE CASCADE);
```

Se considera baza de date populata cu urmatoarele inregistrari:

Tabela studenti:

Marca	Nume	An	Grupa	Media_generala
111111	Rosu Alina	1	1	NULL
222222	Vlad Simona	1	2	NULL
333333	Lung Ionut	2	1	NULL

Tabela discipline:

Cod_disciplina	Denumire_disciplina
1	Programare WEB
2	Baze de date
3	Retele neuronale
4	Java

Tabela note:

Marca	Nota	Cod_disciplina
111111	8	1
111111	10	2
222222	4	2
333333	7	3
333333	8	4

```
SET AUTOCOMMIT ON;
INSERT INTO studenti VALUES('111111', 'Rosu Alina', 1,1,NULL);
INSERT INTO studenti VALUES('222222', 'Vlad Simona',1,2, NULL);
INSERT INTO studenti VALUES('333333', 'Lung Ionut', 2,1, NULL);

INSERT INTO discipline VALUES(1, 'Programare WEB');
INSERT INTO discipline VALUES(2, 'Baze de date');
INSERT INTO discipline VALUES(3, 'Retele neuronale');
INSERT INTO discipline VALUES(4, 'Java');

INSERT INTO note VALUES('111111',8, 1);
INSERT INTO note VALUES('111111',10, 2);
INSERT INTO note VALUES('222222',4, 2);
INSERT INTO note VALUES('333333',7, 3);
INSERT INTO note VALUES('333333',8, 4);
```

a). Sa se creeze o procedura care calculeaza media notelor studentilor dintr-un an de studiu (dat ca si parametru).

b). Scrieţi o procedura care calculează media notelor studentilor dintr-un an de studiu (dat ca si parametru), pentru studenţii integralişti.

c). Sa se creeze o procedura care calculeaza media notelor unui student a carui marca este data ca si parametru si updateaza aceasta medie in tabela studenti. Se vor trata exceptiile care pot sa apara, abandonand operatia in curs si afisand un mesaj de eroare.

d). Sa se creeze o procedura care permite introducerea unei noi note pentru un student (marca studentului, codul disciplinei si noua nota fiind transmise ca si parametri) si recalcularea mediei studentului respectiv in tabela studenti. Se vor trata exceptiile care pot sa apara, abandonand operatia in curs si afisand un mesaj de eroare.