Proiect Elemente Avansate de Baze de Date

Lanţ de magazine

Student: Moţ Carina-Elena

Grupa: 1

Am dezvoltat acest proiect pe baza proiectului de anul trecut, din cadrul materiei „Baze de date”. În cadrul acestor proiecte vom putea regăsi activitatea din cadrul unui lanț de magazine. Acest lanț de magazine este compus din magazinele din ERA, Carrefour, Auchan, Lidl, Kaufland, Casa Rusu și Noriel, care își desfășoară activitatea în mai multe județe ale țării. Fiecare magazin are un număr de angajați, care dețin o anumită funcție și un număr de clienți fideli.

În cadrul acestui proiect am modificat tabela magazine, pentru a adăuga un atribut de tip CLOB(Character Large Object), cu ajutorul căruia câmpul denumire din cadrul tabelei magazine va beneficia de o limită mai mare de caractere.

CREATE TABLE MAGAZINE(

Nr_inreg_mag NUMBER(6) PRIMARY KEY,

Denumire CLOB NOT NULL,

Adresa VARCHAR2(20),

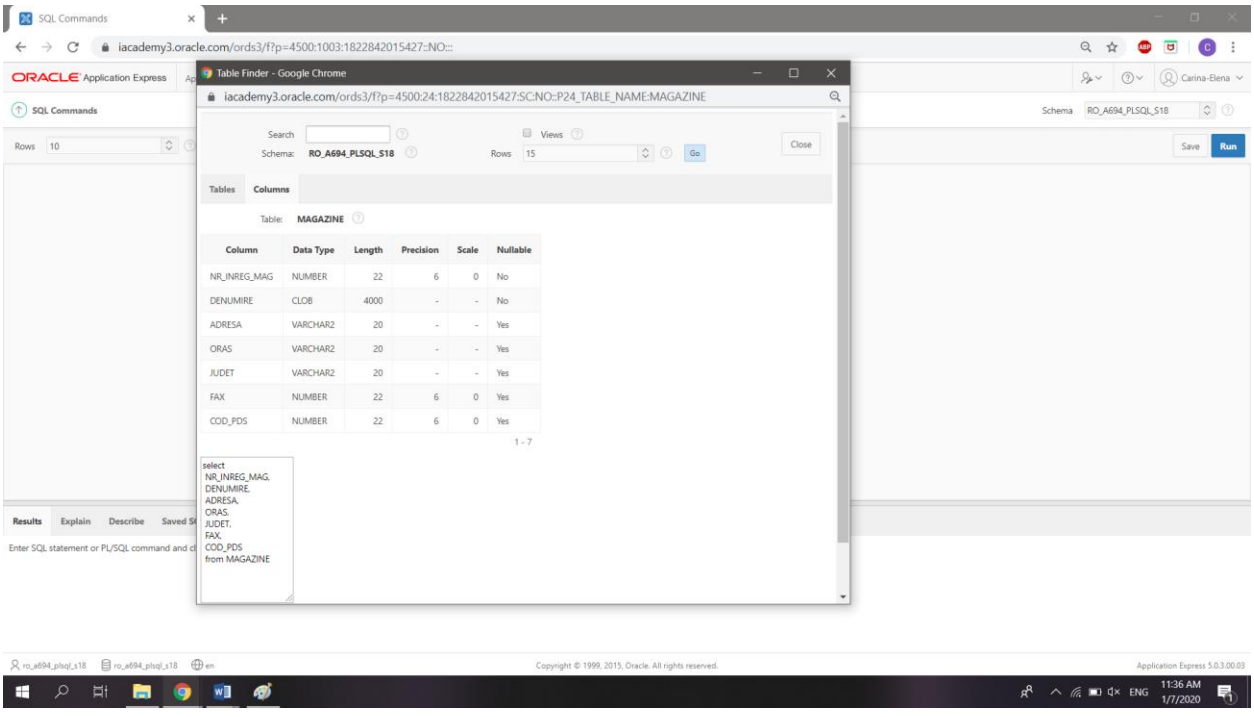
Oras VARCHAR2(20),

Judet VARCHAR2(20),

Fax NUMBER(6),

Cod_pds NUMBER(6) REFERENCES PRODUSE(Cod)

);



Am creat o tabelă ang care va conţine datele angajaţilor care au fost şterşi din baza de date.

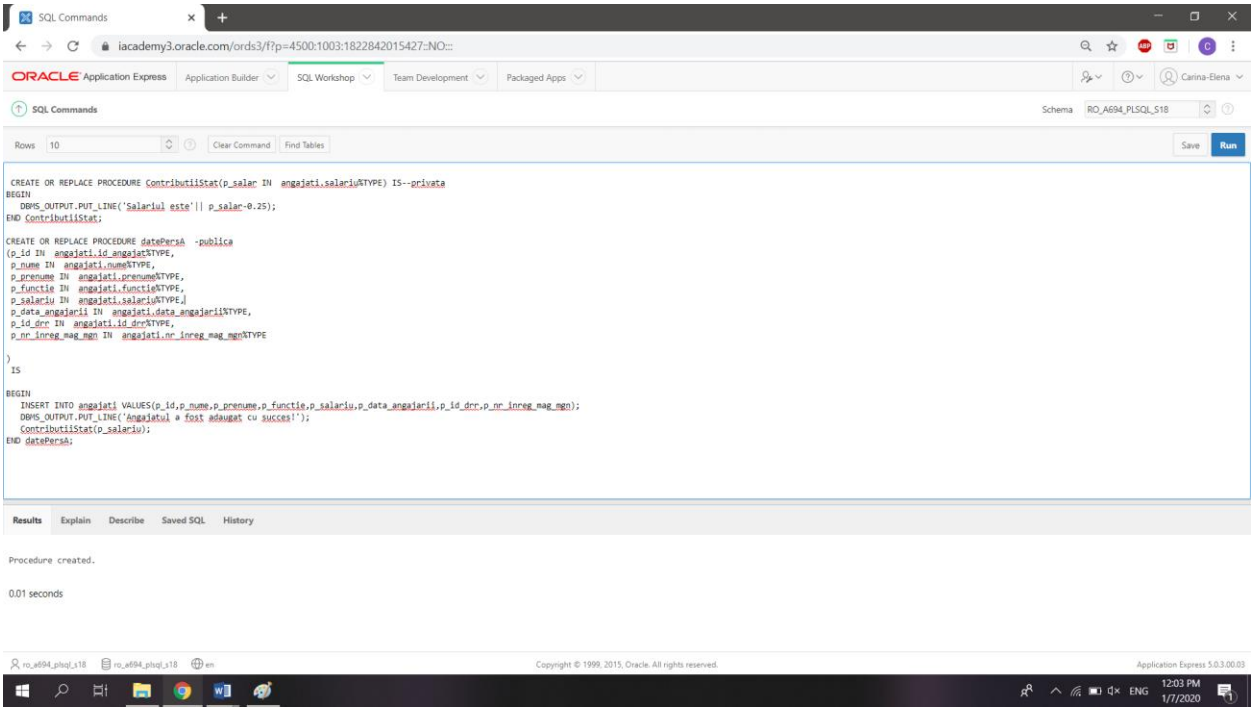
```
Create table ang(
id_angajat INT PRIMARY KEY,
nume VArchar(50),
 prenume Varchar(50),
 telefon INT,
 email varchar(50),
 magazin varchar(50) );
```

Proceduri:

1. Procedura numită datePersA este una dintre procedurile publice existente în acest proiect şi are ca scop inserarea noilor angajaţi, iar procedura privată ContributiiStat are ca scop calcularea sumei constribuţiei pe care fiecare angajat este obligat să o dea.

```
CREATE OR REPLACE PROCEDURE ContributiiStat(p_salar IN  angajati.salariu%TYPE) IS-
-privata
BEGIN
   DBMS_OUTPUT.PUT_LINE('Salariul este'|| p_salar-0.25);
END ContributiiStat;

CREATE OR REPLACE PROCEDURE datePersA  -publica
(p_id IN  angajati.id_angajat%TYPE,
 p_nume IN  angajati.nume%TYPE,
 p_prenume IN  angajati.prenume%TYPE,
 p_functie IN  angajati.functie%TYPE,
 p_salariu IN  angajati.salariu%TYPE,
```

```
 p_data_angajarii IN  angajati.data_angajarii%TYPE,
 p_id_drr IN  angajati.id_drr%TYPE,
 p_nr_inreg_mag_mgn IN  angajati.nr_inreg_mag_mgn%TYPE)
 IS
BEGIN
   INSERT INTO angajati
VALUES(p_id,p_nume,p_prenume,p_functie,p_salariu,p_data_angajarii,p_id_drr,p_nr_inreg_mag_mgn);
   DBMS_OUTPUT.PUT_LINE('Angajatul a fost adaugat cu succes!');
   ContributiiStat(p_salariu);
END datePersA;
```



2. Procedura publica strgereA va efectua stergerea angjatului al cărui id corespunde cu parametrul transmis, în timp ce procedura privată log_angg va insera în tabela ang, id-ul angajatului care a fost șters din baza de date.

CREATE OR REPLACE PROCEDURE log_angg(p_id IN angajati.id_angajat%TYPE)--privata

IS BEGIN

INSERT INTO ang(id_angajat) VALUES(p_id);

END log_angg;

CREATE OR REPLACE PROCEDURE stergereA(p_id_angajat in angajati.id_angajat%TYPE)--publica

IS

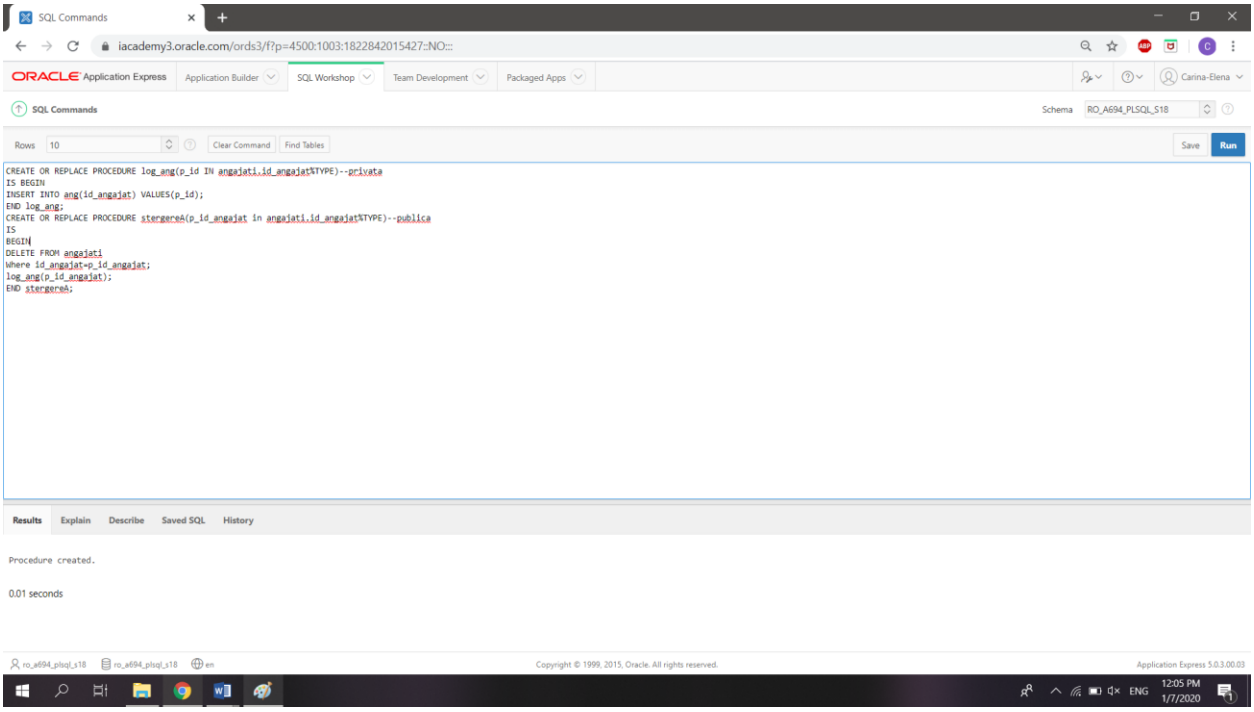
BEGIN

DELETE FROM angajati

Where id_angajat=p_id_angajat;

log_angg(p_id_angajat);

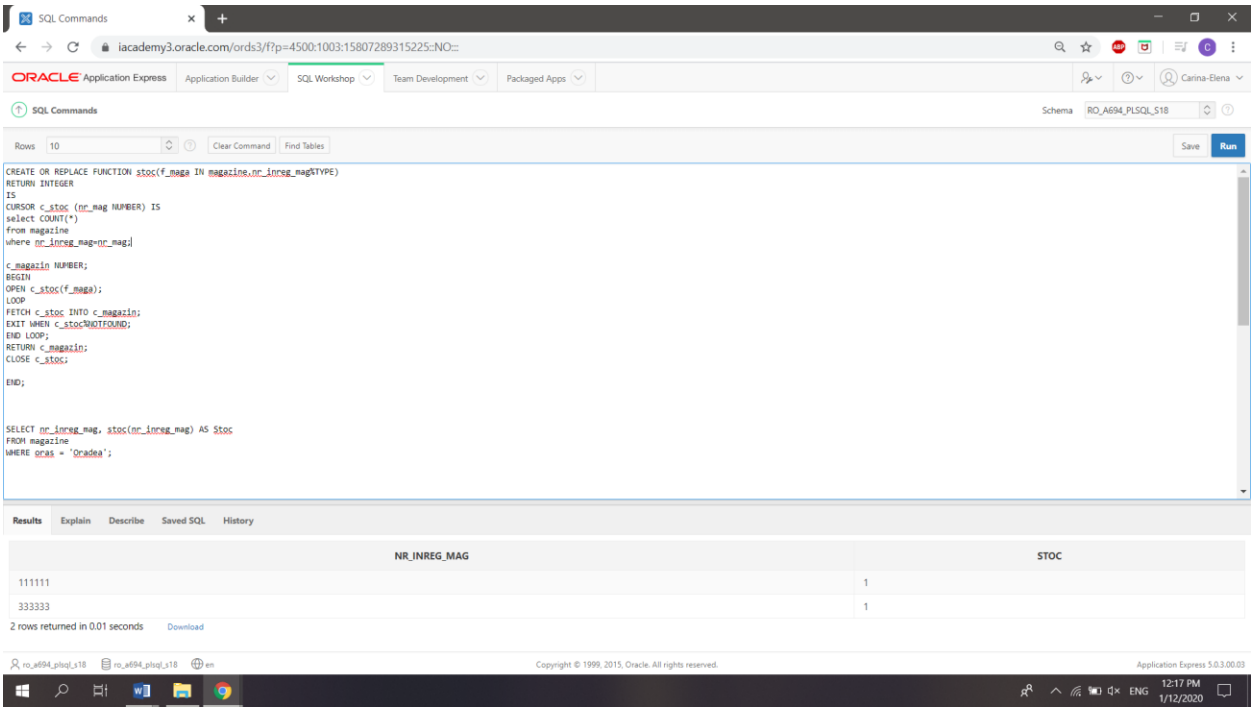
END stergereA;



Funcţii:

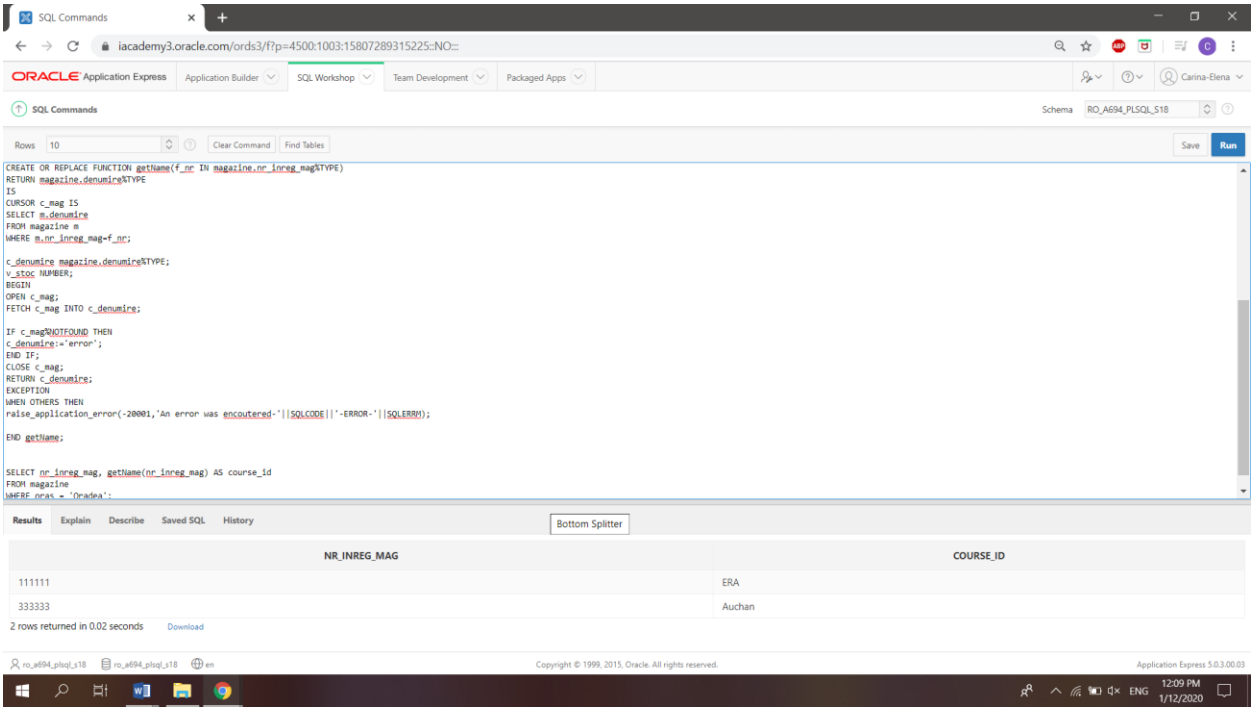
Funcţia privată stoc are rolul de a returna stocul fiecărui magazin în parte, în timp ce funcţia publică getName, are rol în a regăsi magazinul care are id-ul egal cu id-ul furnizat.

```
CREATE OR REPLACE FUNCTION stoc(f_maga IN magazine.nr_inreg_mag%TYPE)

RETURN INTEGER

IS

CURSOR c_stoc (nr_mag NUMBER) IS

select COUNT(*)

from magazine

where nr_inreg_mag=nr_mag;


c_magazin NUMBER;

BEGIN

OPEN c_stoc(f_maga);

LOOP

FETCH c_stoc INTO c_magazin;

EXIT WHEN c_stoc%NOTFOUND;

END LOOP;

RETURN c_magazin;

CLOSE c_stoc;


END;
```




```
CREATE OR REPLACE FUNCTION getName(f_nr IN magazine.nr_inreg_mag%TYPE)
RETURN magazine.denumire%TYPE
IS
CURSOR c_mag IS
SELECT m.denumire
FROM magazine m
WHERE m.nr_inreg_mag=f_nr;

c_denumire magazine.denumire%TYPE;
v_stoc NUMBER;
BEGIN
OPEN c_mag;
FETCH c_mag INTO c_denumire;
```

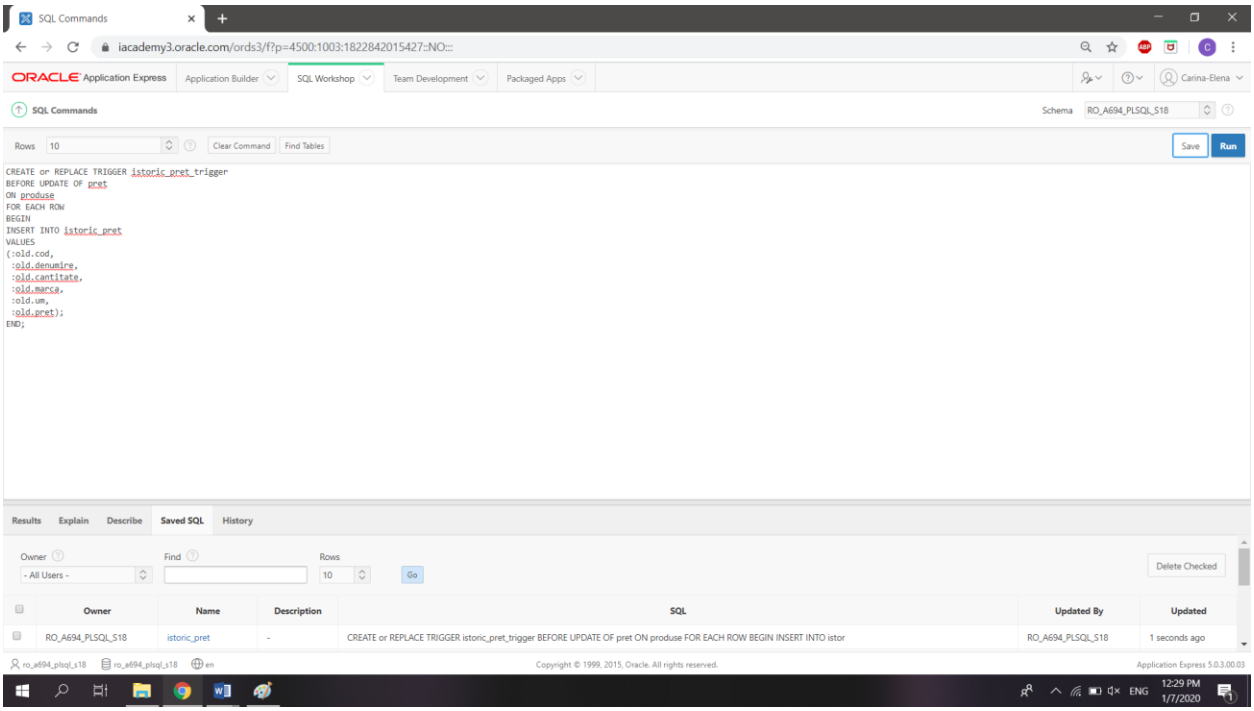
```
IF stoc(f_nr)>0 THEN

DBMS_OUTPUT.PUT_LINE("DA");

END IF;


IF c_mag%NOTFOUND THEN

c_denumire:='error';

END IF;

CLOSE c_mag;

RETURN c_denumire;

EXCEPTION

WHEN OTHERS THEN

raise_application_error(-20001,'An error was encoutered-'||SQLCODE||'-ERROR-'||SQLERRM);


END getName;
```



Declanşatori:

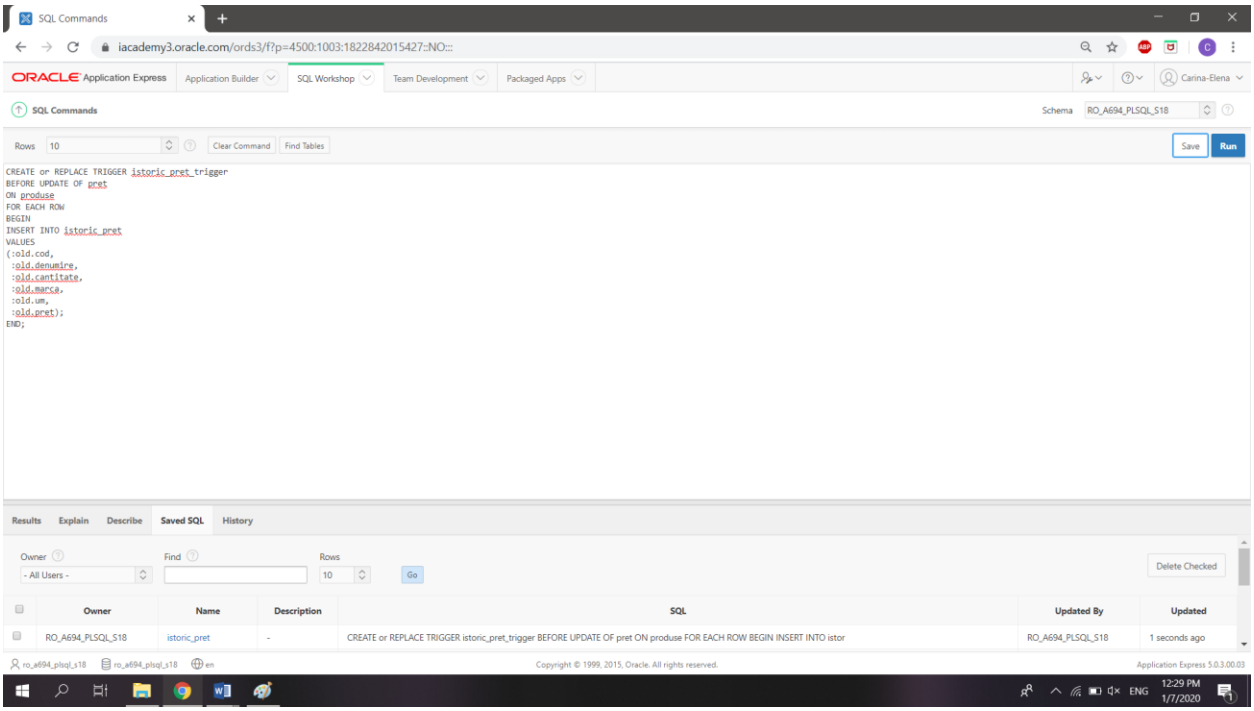
1. Acest declanşator are rolul de a insera în tabela istoric vechiul preţ, inaintge ca acesta să fie actualizat.

```
CREATE or REPLACE TRIGGER istoric_pret_trigger

BEFORE UPDATE OF pret

ON produse

FOR EACH ROW

BEGIN

INSERT INTO istoric_pret

VALUES

(:old.cod,

 :old.denumire,

 :old.cantitate,

 :old.marca,

 :old.um,

 :old.pret);

END;
```



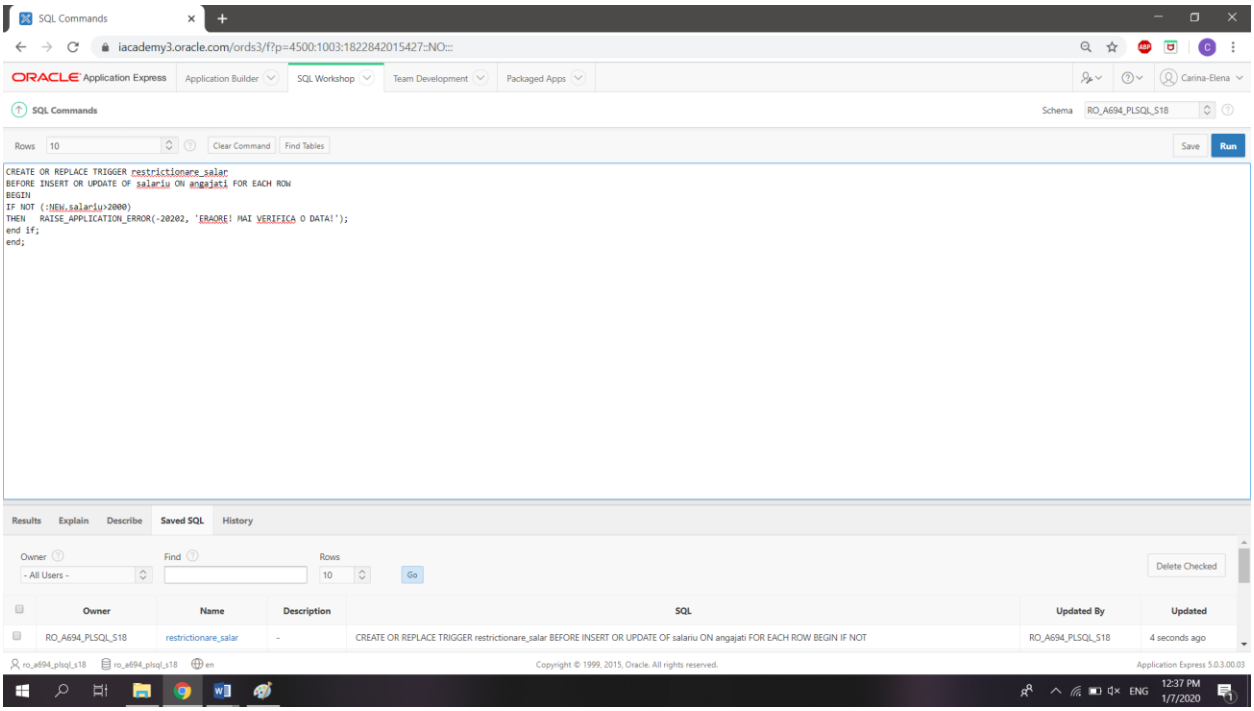
2. Acest declanşator va afişa vechiul salariu, noul salariu şi diferenţa dintre acestea înainte ca în tabela angajaţi să se facă modificările DML.

```
CREATE OR REPLACE TRIGGER salarii
 BEFORE DELETE OR INSERT OR UPDATE ON angajati
 FOR EACH ROW
WHEN (new.id_angajat > 0)
DECLARE
   sal_diff number;
BEGIN
   sal_diff  := :new.salariu  - :old.salariu;
   dbms_output.put('Old salary: ' || :old.salariu);
   dbms_output.put('  New salary: ' || :new.salariu);
   dbms_output.put_line('  Difference ' || sal_diff);
END;
```



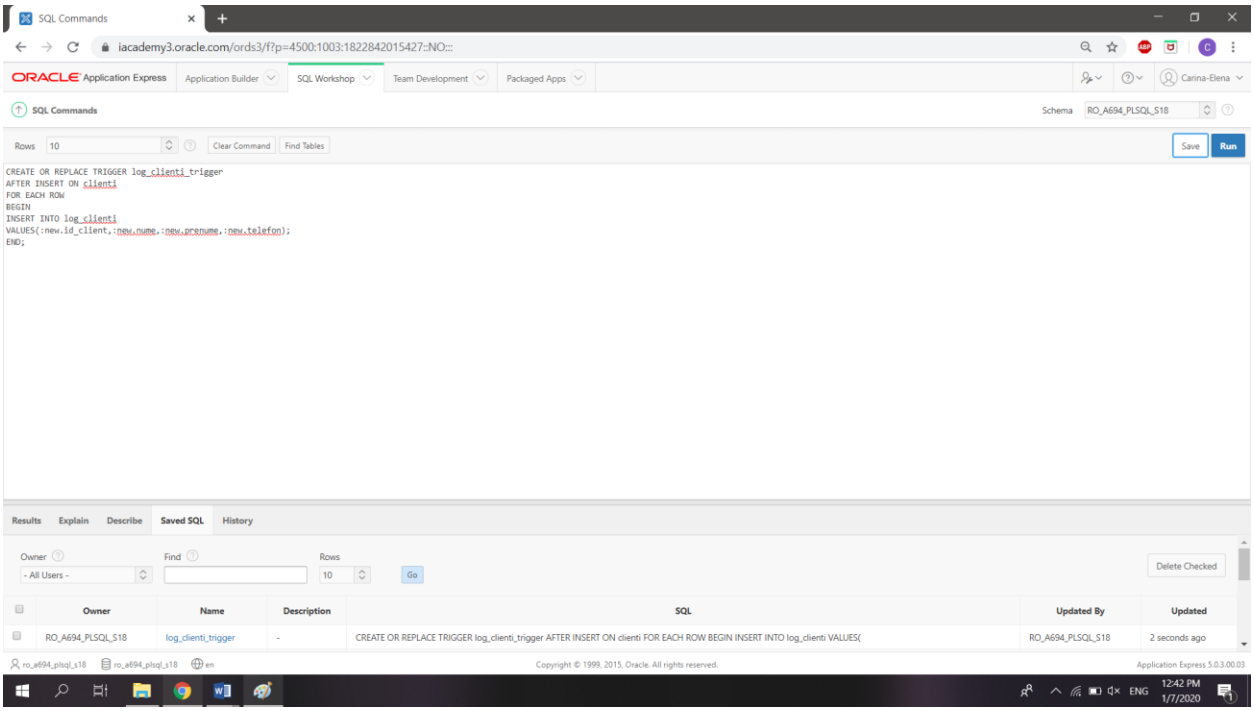
3. Acest declanşator face ca limita inferioară a salariului să fie de 2000 de lei, iar dacă această condiţie nu este îndeplinită, se va declanşa o eroare.

```
CREATE OR REPLACE TRIGGER restrictionare_salar

BEFORE INSERT OR UPDATE OF salariu ON angajati FOR EACH ROW

BEGIN

IF NOT (:NEW.salariu>2000)

THEN   RAISE_APPLICATION_ERROR(-20202, 'ERAORE! MAI VERIFICA O DATA!');

end if;

end;
```



4. Acest declanşator va insera în tabela log_clienţi id-ul, numele, prenumele şi telefonul fiecărui client după fiecare inserare a unui client nou în tabela clienţi.

```
CREATE OR REPLACE TRIGGER log_clienti_trigger

AFTER INSERT ON clienti

FOR EACH ROW

BEGIN

INSERT INTO log_clienti

VALUES(:new.id_client,:new.nume,:new.prenume,:new.telefon);

END;
```



Secvenţe:

1.

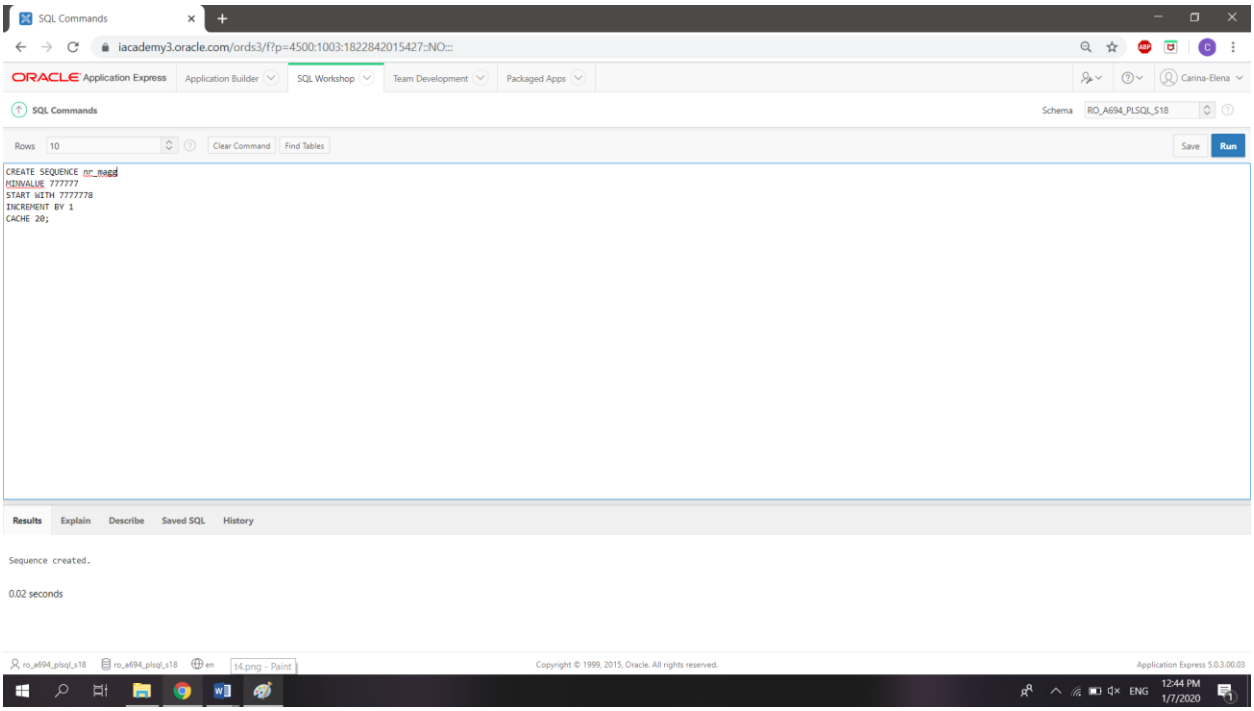
CREATE SEQUENCE nr_magg

MINVALUE 777777

START WITH 7777778

INCREMENT BY 1

CACHE 20;



2.

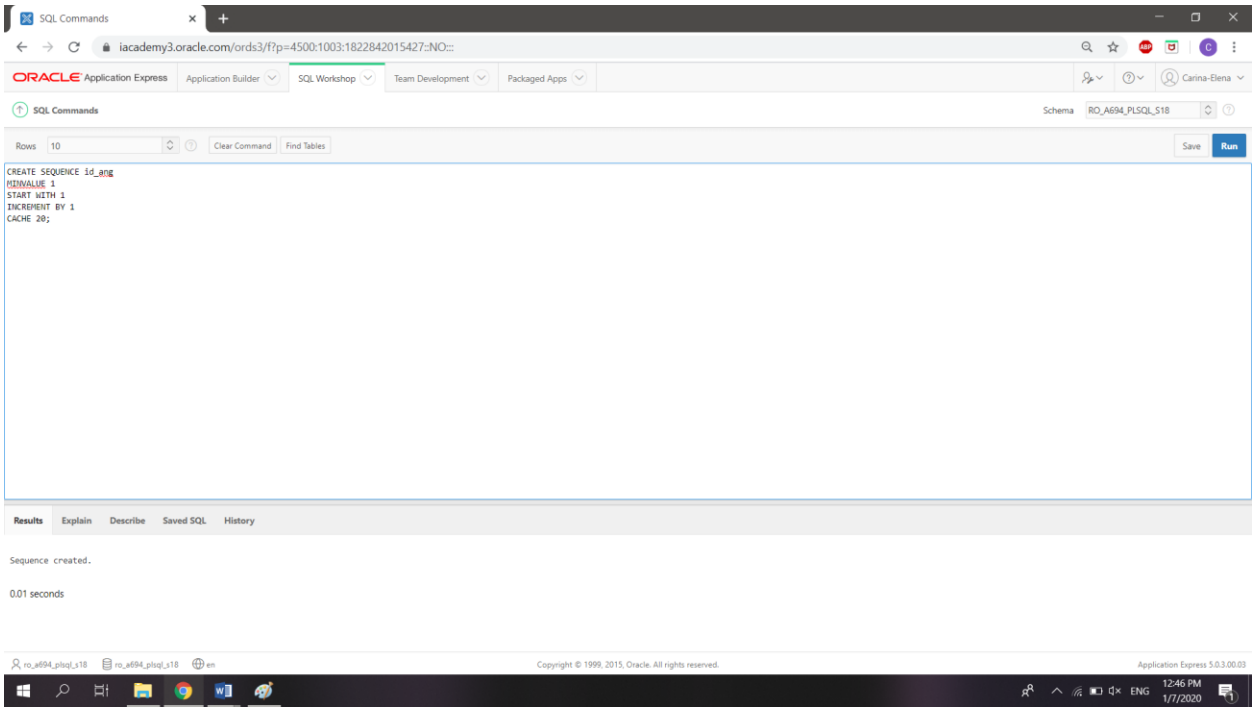
CREATE SEQUENCE id_ang

MINVALUE 1

START WITH 1

INCREMENT BY 1

CACHE 20;



3.

CREATE SEQUENCE id_clienti

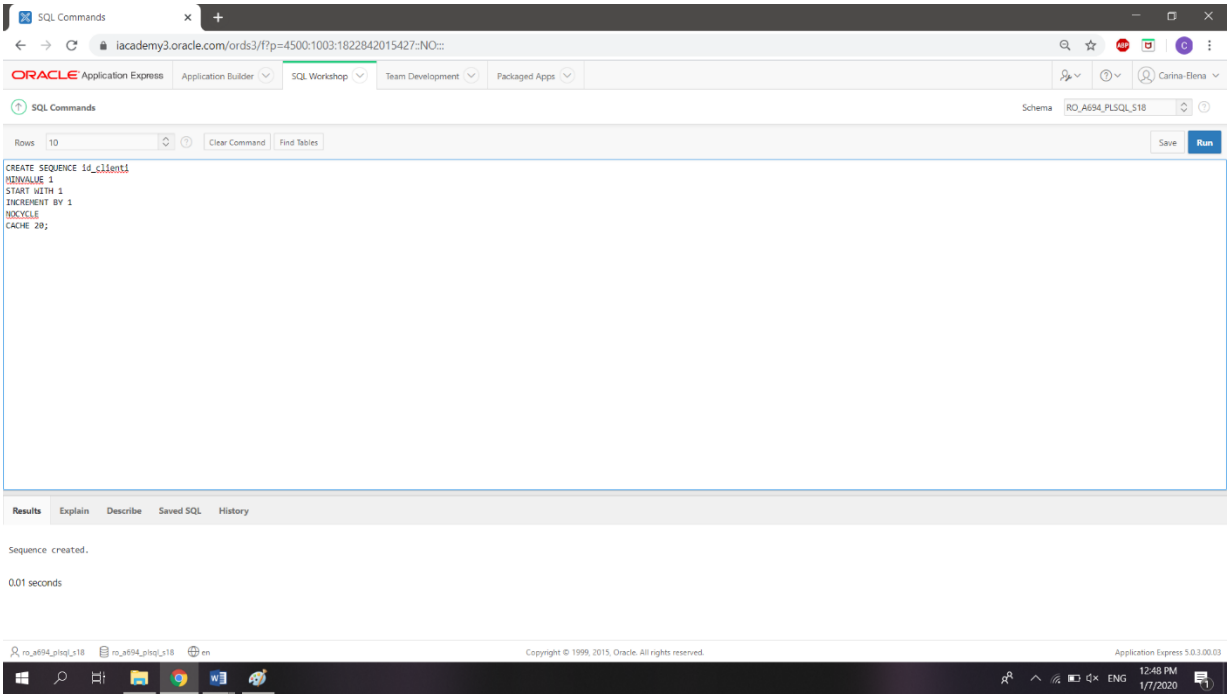
MINVALUE 1

START WITH 1

INCREMENT BY 1

NOCYCLE

CACHE 20;



Package:

CREATE OR REPLACE PACKAGE proiect AS

PROCEDURE datePersA(p_id IN angajati.id_angajat%TYPE, p_nume IN angajati.nume%TYPE, p_prenume IN angajati.prenume%TYPE, p_functie IN angajati.functie%TYPE, p_salariu IN angajati.salariu%TYPE, p_data_angajarii IN angajati.data_angajarii%TYPE, p_id_drr IN angajati.id_drr%TYPE, p_nr_inreg_mag_mgn IN angajati.nr_inreg_mag_mgn%TYPE);

PROCEDURE stergereA(p_id_angajat in angajati.id_angajat%TYPE);

FUNCTION getName(f_nr IN magazine.nr_inreg_mag%TYPE) RETURN magazine.denumire%TYPE;

END proiect;

CREATE OR REPLACE PACKAGE BODY proiect AS

PROCEDURE ContributiiStat(p_salar IN angajati.salariu%TYPE) IS

BEGIN

```
   DBMS_OUTPUT.PUT_LINE('Salariul este'|| p_salar-0.25);

END ContributiiStat;


PROCEDURE datePersA(p_id IN  angajati.id_angajat%TYPE, p_nume IN
angajati.nume%TYPE, p_prenume IN  angajati.prenume%TYPE, p_functie IN
angajati.functie%TYPE, p_salariu IN  angajati.salariu%TYPE, p_data_angajarii IN
angajati.data_angajarii%TYPE, p_id_drr IN  angajati.id_drr%TYPE, p_nr_inreg_mag_mgn IN
angajati.nr_inreg_mag_mgn%TYPE)

 IS

BEGIN

   INSERT INTO angajati
VALUES(p_id,p_nume,p_prenume,p_functie,p_salariu,p_data_angajarii,p_id_drr,p_nr_inreg_m
ag_mgn);

   DBMS_OUTPUT.PUT_LINE('Angajatul a fost adaugat cu succes!');

   ContributiiStat(p_salariu);

END datePersA;


PROCEDURE log_angg(p_id IN angajati.id_angajat%TYPE)

IS BEGIN

INSERT INTO ang(id_angajat) VALUES(p_id);

END log_angg;

PROCEDURE stergereA(p_id_angajat in angajati.id_angajat%TYPE)

IS

BEGIN

DELETE FROM angajati

Where id_angajat=p_id_angajat;

log_angg(p_id_angajat);
```

```
END stergereA;
CREATE OR REPLACE FUNCTION stoc(f_maga IN magazine.nr_inreg_mag%TYPE)
RETURN INTEGER
IS
CURSOR c_stoc (nr_mag NUMBER) IS
select COUNT(*)
from magazine
where nr_inreg_mag=nr_mag;

c_magazin NUMBER;
BEGIN
OPEN c_stoc(f_maga);
LOOP
FETCH c_stoc INTO c_magazin;
EXIT WHEN c_stoc%NOTFOUND;
END LOOP;
RETURN c_magazin;
CLOSE c_stoc;
END;

FUNCTION getName(f_nr IN magazine.nr_inreg_mag%TYPE)
RETURN magazine.denumire%TYPE
IS
CURSOR c_mag IS
SELECT m.denumire
```

```
FROM magazine m
WHERE m.nr_inreg_mag=f_nr;


c_denumire magazine.denumire%TYPE;
v_stoc NUMBER;
BEGIN
OPEN c_mag;
LOOP
FETCH c_mag INTO c_denumire;
EXIT WHEN c_mag%NOTFOUND;
END LOOP;
CLOSE c_mag;
RETURN c_denumire;
END getName;
END proiect;
```