PROIECT

SISTEME DE GESTIUNE A

BAZELOR DE DATE

IONESCU ALEXANDRU

GRUPA 241

UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

SPECIALIZAREA INFORMATICĂ

PROFESOR COORDONATOR

**NATALIA GABRIELA MOANGA**

Cuprins

[Studiu de caz – Cerinta 1 1](#_Toc92201315)

[Entități 1](#_Toc92201316)

[Relaţii 2](#_Toc92201317)

[Atribute 3](#_Toc92201318)

[Diagrama entitate – relaţie – Cerinta 2 7](#_Toc92201319)

[Diagrama conceptuală – Cerinta 3 8](#_Toc92201320)

[Schemele relaţionale 9](#_Toc92201321)

[Diagrama entitate-relatie, cu atribute 10](#_Toc92201322)

[Crearea bazei de date si inserarea informatiilor – Cerintele 4 si 5 11](#_Toc92201323)

[Codul pentru creare si inserare 11](#_Toc92201324)

[Rezultatele rularii codului 21](#_Toc92201325)

[Normalizarea FN1-FN3 24](#_Toc92201326)

[Tabelul ANGAJAT 24](#_Toc92201327)

[Forma normala 1 (FN1) 24](#_Toc92201328)

[Forma normala 2 (FN2) 25](#_Toc92201329)

[Forma normala 3 (FN3) 26](#_Toc92201330)

[Tabelul FACTURA 27](#_Toc92201331)

[Celelalte tabele 28](#_Toc92201332)

[Cerintele 6 – 13 28](#_Toc92201333)

[Cerinta 6 28](#_Toc92201334)

[Cerinta 7 30](#_Toc92201335)

[Cerinta 8 32](#_Toc92201336)

[Cerinta 9 35](#_Toc92201337)

[Cerinta 10 38](#_Toc92201338)

[Cerinta 11 40](#_Toc92201339)

[Cerinta 12 42](#_Toc92201340)

[Cerinta 13 43](#_Toc92201341)

COMPANIE TAXI

# Studiu de caz – Cerinta 1

Acest capitol se referă la proiectarea unui model de date ce furnizează informații despre o companie de taxi și desfășurarea acesteia.

Vom prezenta modelul de date, restricţiile pe care trebuie să le respecte şi vom încerca să construim diagrama E/R corespunzătoare. Vom considera, în abordarea iniţială, anumite situaţii care nu sunt optime, în sensul că pot genera redundanţă, anomalii la reactualizări sau nu permit rezolvarea anumitor interogări asupra modelului. Vom încerca să arătăm care sunt deficienţele modelului, situaţiile care le-au generat şi cum pot fi corectate (parţial sau total) anomaliile respective.

Modelul de date va gestiona informații legate de organizarea și funcționarea unei companii de taximetrie. Aceasta are mai multe tipuri de angajați, care se ocupa atât de conducerea mașinilor, cât și de gestionarea comenzilor și plăților. Există informații atât despre facturi, cât și despre șoferi și performanțele acestora, dar și mai multe detalii despre orice comanda care s-a desfășurat. Toate mașinile sunt de asemenea înregistrate.

Modelul de date respectă anumite restricţii de funcţionare.

* Un angajat poate fi ori **sofer**, ori **dispecer**.
* Un șofer poate conduce o singură mașină la un moment dat, iar o mașină poate fi condusă doar de un șofer.
* Un client primește un discount sau nici un discount, în funcție de notă.
* Doar un dispecer poate emite o factură. Una sau mai multe facturi au un discount sau nici un discount.
* O cursă are o factură și un set de detalii. O mașină este folosită la mai multe curse.

## Entități

Pentru modelul de date referitor la compania taxi, structurile ANGAJAT, CURSA, CLIENT, FACTURA, DISCOUNT, MASINA, DETALII\_CURSA, ISTORIC\_SOFER, LOCATII sunt entitățile modelului de date.

Vom prezenta entităţile modelului de date, dând o descriere completă a fiecăreia. De asemenea, pentru fiecare entitate se vă preciza cheia primară.

Toate entităţile care vor fi prezentate sunt independente, cu excepţia subentităţilor SOFER, RECEPTIONIST.

ANGAJAT = (Persoană) încadrată într-un loc de muncă. Aceasta poate fi atât șofer, care conduce o mașină, are un istoric, sau este dispecer. Cheia primară a entității este cod\_angajat.

SOFER = Persoană calificată pentru a conduce un autovehicul, conduce o masina. Este un angajat. Cheia primara a entității este cod\_sofer.

DISPECER = Tehnician sau sistem automat care urmărește, coordonează și reglementează operativ mersul producției dintr-o întreprindere. Este un angajat. Cheia primară a entității este cod\_dispecer.

CURSA = Drum parcurs spre o anumită destinație, la preferința clientului, care este facturată, realizată de un șofer cu o mașină. Cheia primară a entității este cod\_cursa.

CLIENT = Persoană care achiziționează o cursă, într-o mașină de-a companiei, condusă de un șofer al companiei. Acesta are un scor calculat în funcție de notele primite Cheia primară a entității este cod\_client.

FACTURA = Entitate în care se memorează data când s-a desfășurat cursa, prețul acesteia. Cheia primară a entității este cod\_factura.

DISCOUNT = Entitate care stochează discount-ul în funcție de scorul clientului. Cheia primară a entității este cod\_discount.

VEHICUL = Autoturism folosit de șofer pentru a transporta clientul la destinație, pentru a completa cursele. Cheia primară a entității este cod\_vehicul.

FACTURARE = Detalii despre facturare, suma plătită, tipul plății, TVA. Cheia primară este cod\_factura.

DETALII\_CURSA = Entitate care conține detalii despre cursă precum cheia șoferului,cheia mașinii, data, nota acordată șoferului. Cheia primară este cod\_detalii\_cursa.

ISTORIC\_SOFER = O entitate în care sunt stocate numărul de curse ale șoferului, ratingul calculat în funcție de curse și salariul. Cheia primară a entității este cod\_sofer.

LOCATII = O entitate în care sunt stocate localitatea și județul în care un șofer are licență pentru prestarea serviciilor de taxi. Cheia primară a entității este cod\_locatie.

## Relaţii

Vom prezenta relaţiile modelului de date, dând o descriere completă a fiecăreia. De fapt, denumirile acestor legături sunt sugestive, reflectând conţinutul acestora şi entităţile pe care le leagă. Pentru fiecare relaţie se vă preciza cardinalitatea minimă şi maximă.

SOFER\_face\_CURSA = relație care leagă entitățile SOFER și CURSA, reflectând legătura dintre acestea (ce cursă face un anumit șofer). Ea are cardinalitatea minimă 1:1 (un șofer trebuie să realizeze cel puțin o cursă și o cursă trebuie făcută de un șofer) și cardinalitatea maximă 1:n (un șofer poate face mai multe curse, iar o cursă poate fi făcută de un singur șofer).

SOFER\_are\_ISTORIC\_SOFER = relație care leagă entitățile SOFER și ISTORIC\_SOFER, reflectând legătura dintre acestea (ce istoric are un anumit șofer). Ea are cardinalitatea 1:1, întrucât un șofer poate avea un singur istoric despre el însuși.

VEHICUL\_este\_condus\_de\_SOFER = relație care leagă entitățile VEHICUL și SOFER, reflectând legătura dintre acestea (ce vehicul conduce un anumit șofer). Ea are cardinalitatea 1:1, întrucât un vehicul poate fi condus decât de un șofer.

VEHICUL\_este\_folosit\_la\_CURSA = relație care leagă entitățile VEHICUL și CURSA, reflectând legătura dintre acestea (ce vehicul este folosit la o cursă). Ea are cardinalitatea 1:1, întrucât un vehicul poate fi folosit decât la o cursă.

CURSA\_are\_DETALII\_CURSA = relație care leagă entitățile CURSA și DETALII\_CURSA, reflectând legătura dintre acestea (care sunt detaliile unei anumite curse). Are cardinalitatea 1:1, întrucât o cursă nu poate avea decât un set de detalii.

CURSA\_are\_FACTURA = relație care leagă entitățile CURSA și FACTURA, reflectând legătura dintre acestea (ce factură are o cursă). Are cardinalitatea 1:1, întrucât o cursă este facturată decât o dată.

FACTURA\_are\_DISCOUNT = relație care leagă entitățile FACTURA și DISCOUNT, reflectând legătura dintre acestea (ce discount are o factură). Are cardinalitatea minma 1:0 și cardinalitatea maximă n:1.

CLIENT\_primeste\_DISCOUNT = relație care leagă entitățile CLIENT și DISCOUNT, reflectând legătura dintre acestea (ce discount primește un client). Ea are cardinalitatea minimă 1:0, iar cardinalitatea maximă este 1:1.

CLIENT\_cumpara\_CURSA = relație care leagă entitățile CLIENT și CURSA, reflectând legătura dintre acestea (ce client cumpără o cursă). Ea are cardinalitatea minimă 1:1, iar cardinalitatea maximă este 1:n.

DISPECER\_emite\_FACTURA = relație care leagă entitățile DISPECER și FACTURA, reflectând legătura dintre acestea (ce dispecer a emis o factură). Are cardinalitatea minimă 1:1 și cardinalitatea maximă 1:n.

ANGAJAT\_lucreaza\_in\_LOCATII = relație care leagă entitățile ANGAJAT și LOCATII, reflectând legătura dintre acestea (în ce locații lucrează angajații). Are cardinalitatea minimă 1:1 și cardinalitatea maximă n:n.

## Atribute

Entitatea independentă CLIENT are ca variabile:

cod\_client = variabilă de tip întreg, de lungime maximă 6, care reprezintă codul clientului.

nume = variabilă de tip caracter, de lungime maximă 25, care reprezintă numele clientului.

prenume = variabilă de tip caracter, de lungime maximă 25, care reprezintă prenumele clientului.

nr\_telefon = variabilă de tip caracter, de lungime maximă 12, care reprezintă numărul de telefon al clientului.

dată\_naștere = variabilă de tip dată calendaristică, care reprezintă data naşterii clientului respectiv.

apelativ = variabilă de tip caracter, de lungime maximă 5, care reprezintă apelativul clientului (Dl., Dna., Dra., etc.).

nota = variabilă de tip întreg, de lungime maximă 2, care reprezintă codul clientului.

Entitatea independentă ANGAJAT are ca variabile:

cod\_angajat = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul angajatului.

nume = variabilă de tip caracter, de lungime maximă 25, care reprezintă numele angajatului.

prenume = variabilă de tip caracter, de lungime maximă 25, care reprezintă prenumele angajatului.

nr\_telefon = variabilă de tip caracter, de lungime maximă 12, care reprezintă numărul de telefon al angajatului.

data\_angajare = variabilă de tip dată calendaristică, care reprezintă data angajării angajatului.

data\_nastere = variabilă de tip dată calendaristică, care reprezintă data naşterii angajatului respectiv.

Subentitatea SOFER are ca atribute:

cod\_angajat = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul șoferului.

numar\_masina = variabilă de tip caracter, de lungime maximă 10, care reprezintă codul mașinii (numărul de înmatriculare). Atributul trebuie să corespundă la o valoare a cheii primare din tabelul VEHICUL.

Subentitatea DISPECER are ca atribute:

cod\_angajat = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul dispecerului.

dispecerat = variabilă de tip caracter, de lungime maximă 15, care reprezintă numele dispeceratului unde este alocat.

Entitatea independentă MASINA are ca atribute:

cod\_masina = variabilă de tip caracter, de lungime maximă 10, care reprezintă codul mașinii (numărul de înmatriculare).

data\_achizitionare = variabilă de tip dată calendaristică, care reprezintă data cumpărării mașinii.

data\_revizie\_urmatoare = variabilă de tip dată calendaristică, care reprezintă data următoarei revizii.

cod\_sofer = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul șoferului. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul SOFER.

marca = variabilă de tip caracter, de lungime maximă 20, care reprezintă marca mașinii.

model = variabilă de tip caracter, de lungime maximă 20, care reprezintă modelul mașinii.

Entitatea independentă DISCOUNT are ca atribute:

cod\_discount = variabilă de tip întreg, de lungime maximă 2, care reprezintă codul discount-ului, dar și valoarea acestuia, în procente.

nota\_client = variabilă de tip întreg, de lungime maximă 2, care reprezintă nota pentru care se aplică discount-ul.

Entitatea independentă CURSA are ca atribute:

cod\_cursa = variabilă de tip întreg, de lungime maximă 6, care reprezintă codul cursei.

cod\_masina = variabilă de tip caracter, de lungime maximă 10, care reprezintă codul mașinii (numărul de înmatriculare). Atributul trebuie să corespundă la o valoare a cheii primare din tabelul MASINA.

cod\_sofer = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul șoferului. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul SOFER.

cod\_client = variabilă de tip întreg, de lungime maximă 6, care reprezintă codul clientului. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul CLIENT.

adresa\_client = variabilă de tip caracter, de lungime maximă 25, care reprezintă adresa de unde este preluat clientul.

destinatie = variabilă de tip caracter, de lungime maximă 25, care reprezintă adresa unde este lăsat clientul.

cod\_locatie = variabilă de tip întreg, de lungime maximă 4, care reprezintă codul locației unde s-a desfășurat cursa. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul LOCATII.

Entitatea DETALII\_CURSA are ca atribute:

cod\_cursa = variabilă de tip întreg, de lungime maximă 6, care reprezintă codul cursei. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul CURSA.

data\_cursa = variabilă de tip dată calendaristică, care reprezintă data efectuării cursei.

nota\_sofer = variabilă de tip întreg, de lungime maximă 2, care reprezintă nota șoferului.

nota\_client = variabilă de tip întreg, de lungime maximă 2, care reprezintă nota clientului.

Entitatea independentă FACTURA are ca atribute:

cod\_factura = variabilă de tip întreg, de lungime maximă 6, care reprezintă codul facturii.

cod\_dispecer = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul dispecerului. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul DISPECER.

cod\_sofer = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul șoferului. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul SOFER.

pret = variabilă de tip float, de lungime maximă 4, cu maxim 2 zecimale, care reprezintă costul cursei.

Entitatea independentă ISTORIC\_SOFER are ca atribute:

cod\_sofer = variabilă de tip întreg, de lungime maximă 6, care reprezintă codul șoferului. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul SOFER.

nota = variabilă de tip float, de lungime maximă 2, cu maxim 2 zecimale, care reprezintă media notelor șoferului.

numar\_curse = variabilă de tip întreg, de lungime maximă 5, care reprezintă numărul de curse efectuate de șofer.

Entitatea independentă LOCATII are ca atribute:

cod\_locatie = variabilă de tip întreg, de lungime maximă 4, care reprezintă codul locației unde s-a desfășurat cursa.

localitate = variabilă de tip caracter, de lungime maximă 20, care reprezintă numele localității unde se desfășoară cursa.

judet = variabilă de tip caracter, de lungime maximă 20, care reprezintă numele județului unde se desfășoară cursa.

# Diagrama entitate – relaţie – Cerinta 2

# Diagrama conceptuală – Cerinta 3

Schemele relaţionale corespunzătoare diagramei conceptuale sunt următoarele:

ANGAJAT(cod\_angajat#, nume, prenume, nr\_telefon, data\_nastere, data\_angajare, tip\_angajat, salariu, numar\_masina, dispecerat)

ISTORIC\_SOFER(cod\_sofer#, nota, numar\_curse)

CLIENT(cod\_client#, nume, prenume, nr\_telefon, apelativ, data\_nastere, nota)

DISCOUNT(cod\_discount#, nota\_client)

CURSA(cod\_cursa#, cod\_masina, cod\_sofer, cod\_client, adresa\_client, destinatie, cod\_locatie)

DETALII\_CURSA(cod\_cursa#, data\_cursa, nota\_sofer, nota\_client, cod\_masina)

FACTURA(cod\_factura#, cod\_dispecer, cod\_cursa, pret)

MASINA(cod\_masina#, data\_achizitionare, data\_revizie\_urm, marca, model)

LOCATII(cod\_locatie#, localitate, judet)

LUCREAZA\_IN(cod\_angajat#, cod\_locatie#)

# Diagrama entitate-relatie, cu atribute

Graphical user interface, application

Description automatically generated

# Crearea bazei de date si inserarea informatiilor – Cerintele 4 si 5

## Codul pentru creare si inserare

CREATE TABLE CLIENT(

cod\_client number(6) constraint pkey\_clnt1 primary key,

nume varchar2(25) not null,

prenume varchar2(25) not null,

nr\_telefon varchar2(12) not null,

apelativ varchar2(5),

data\_nastere date not null,

nota number(3,1) not null);

CREATE TABLE masina(

cod\_masina varchar2(10) CONSTRAINT pkey\_mas1 PRIMARY KEY,

data\_achizitionare date not null,

data\_revizie\_urm date not null,

marca varchar2(20) not null,

model varchar2(20) not null);

CREATE TABLE angajat(

cod\_angajat number(5) CONSTRAINT pkey\_ang1 PRIMARY KEY,

nume varchar2(25) not null,

prenume varchar2(25) not null,

nr\_telefon varchar2(12) not null,

tip\_angajat varchar2(25) not null,

data\_nastere date not null,

data\_angajare date not null,

salariu number(6) not null,

numar\_masina varchar2(12),

dispecerat varchar2(25),

CONSTRAINT unic unique (numar\_masina),

CONSTRAINT fkey\_masina FOREIGN KEY (numar\_masina) REFERENCES masina(cod\_masina));

CREATE TABLE locatii(

cod\_locatie number(4) CONSTRAINT pkey\_loc PRIMARY KEY,

localitate varchar2(20) not null,

judet varchar2(20) not null);

CREATE TABLE cursa(

cod\_cursa number(6) CONSTRAINT pkey\_cursa PRIMARY KEY,

cod\_masina varchar2(12) CONSTRAINT fkey\_masina2 REFERENCES angajat(numar\_masina) not null,

cod\_sofer number(5) CONSTRAINT fkey\_sofer2 REFERENCES angajat(cod\_angajat) not null,

cod\_client number(6) CONSTRAINT fkey\_client2 REFERENCES client(cod\_client) not null,

adresa\_client varchar2(35) not null,

destinatie varchar2(35) not null,

cod\_locatie number(4) CONSTRAINT fkey\_loc1 REFERENCES locatii(cod\_locatie) not null);

CREATE TABLE detalii\_cursa(

cod\_cursa number(6) CONSTRAINT pkey\_det\_cursa PRIMARY KEY,

CONSTRAINT fkey\_cod\_cursa FOREIGN KEY (cod\_cursa) REFERENCES cursa(cod\_cursa),

data\_cursa date not null,

nota\_sofer number(2) not null,

nota\_client number(2) not null);

CREATE TABLE istoric\_sofer(

cod\_sofer number(6) CONSTRAINT pkey\_ist\_sof PRIMARY KEY,

CONSTRAINT fkey\_cod\_sof FOREIGN KEY (cod\_sofer) REFERENCES angajat(cod\_angajat),

nota number(4,2) not null,

numar\_curse number(5) not null);

CREATE TABLE discount(

nota\_discount number(2)CONSTRAINT pkey\_disc PRIMARY KEY not null,

cod\_discount number(2));

CREATE TABLE factura(

cod\_factura number(6) CONSTRAINT pkey\_fact PRIMARY KEY,

cod\_dispecer number(5)

CONSTRAINT fkey\_disp REFERENCES angajat(cod\_angajat) not null,

cod\_cursa number(6) CONSTRAINT fkey\_crs REFERENCES cursa(cod\_cursa) not null,

pret number(5,2) not null);

CREATE TABLE lucreaza\_in(

cod\_angajat number(5) CONSTRAINT fkey\_ang references angajat(cod\_angajat),

cod\_locatie number(5) CONSTRAINT fkey\_loc references locatii(cod\_locatie),

CONSTRAINT pk\_compus PRIMARY KEY(cod\_angajat,cod\_locatie));

INSERT INTO masina

VALUES('B 24 TAX', TO\_DATE('1-6-2008','dd-mm-yyyy'), TO\_DATE('20-3-2022','dd-mm-yyyy'),'Dacia','Logan');

INSERT INTO masina

VALUES('B 124 PEL', TO\_DATE('20-11-2010','dd-mm-yyyy'), TO\_DATE('3-1-2022','dd-mm-yyyy'),'Skoda','Octavia');

INSERT INTO masina

VALUES('IF 745 RBE', TO\_DATE('13-12-2014','dd-mm-yyyy'), TO\_DATE('15-12-2021','dd-mm-yyyy'),'Renault','Megane');

INSERT INTO masina

VALUES('B 123 TAX', TO\_DATE('1-10-2020','dd-mm-yyyy'), TO\_DATE('7-5-2022','dd-mm-yyyy'),'Dacia','Logan');

INSERT INTO masina

VALUES('B 167 TAX', TO\_DATE('1-10-2020','dd-mm-yyyy'), TO\_DATE('20-6-2021','dd-mm-yyyy'),'Dacia','Logan');

INSERT INTO masina

VALUES('IS 24 FLV', TO\_DATE('1-5-2010','dd-mm-yyyy'), TO\_DATE('1-5-2022','dd-mm-yyyy'),'Chevrolet','Aveo');

CREATE SEQUENCE ID\_ANG

INCREMENT by 1

START WITH 100

MAXVALUE 1000

NOCYCLE;

INSERT INTO ANGAJAT VALUES(ID\_ANG.NEXTVAL,'Marinescu','Cristian','0742536126','Sofer',TO\_DATE('21-3-1973','dd-mm-yyyy'),TO\_DATE('8-9-2017','dd-mm-yyyy'),3000,'B 24 TAX',NULL);

INSERT INTO ANGAJAT VALUES(ID\_ANG.NEXTVAL,'Petre','Ionel','0723548215','Sofer',TO\_DATE('27-8-1969','dd-mm-yyyy'),TO\_DATE('1-10-2018','dd-mm-yyyy'),3200,'B 124 PEL',NULL);

INSERT INTO ANGAJAT VALUES(ID\_ANG.NEXTVAL,'Pop','Alina','0731446094','Dispecer',TO\_DATE('1-9-1990','dd-mm-yyyy'),TO\_DATE('15-1-2018','dd-mm-yyyy'),2500,NULL,'Titan');

INSERT INTO ANGAJAT VALUES(ID\_ANG.NEXTVAL,'Georgescu','Damian','0732989824','Dispecer',TO\_DATE('23-11-1987','dd-mm-yyyy'),TO\_DATE('16-6-2016','dd-mm-yyyy'),2500,NULL,'Titan');

INSERT INTO ANGAJAT VALUES(ID\_ANG.NEXTVAL,'Oprea','Teohari','0244994900','Dispecer',TO\_DATE('12-12-1988','dd-mm-yyyy'),TO\_DATE('8-2-2020','dd-mm-yyyy'),2500,NULL,'Dristor');

INSERT INTO ANGAJAT VALUES(ID\_ANG.NEXTVAL,'Dragoi','Rebeca','0759330571','Sofer',TO\_DATE('17-7-1978','dd-mm-yyyy'),TO\_DATE('22-11-2017','dd-mm-yyyy'),3200,'IF 745 RBE',NULL);

INSERT INTO ANGAJAT VALUES(ID\_ANG.NEXTVAL,'Marin','Flavius','0749382571','Sofer',TO\_DATE('7-7-1991','dd-mm-yyyy'),TO\_DATE('7-9-2019','dd-mm-yyyy'),3000,'IS 24 FLV',NULL);

INSERT INTO ANGAJAT VALUES(ID\_ANG.NEXTVAL,'Mocanu','Vlad','0756256698','Dispecer',TO\_DATE('10-2-1981','dd-mm-yyyy'),TO\_DATE('8-4-2021','dd-mm-yyyy'),2200,NULL,'Dristor');

INSERT INTO ANGAJAT VALUES(ID\_ANG.NEXTVAL,'Georgescu','Matei','0756854824','Dispecer',TO\_DATE('23-11-1977','dd-mm-yyyy'),TO\_DATE('16-6-2015','dd-mm-yyyy'),2700,NULL,'Titan');

CREATE SEQUENCE ID\_CLIENT

INCREMENT by 1

START WITH 100

MAXVALUE 1000

NOCYCLE;

INSERT INTO CLIENT VALUES(ID\_CLIENT.NEXTVAL,'Martinescu','Vali','0724585435','Dl.',TO\_DATE('27-11-2000','dd-mm-yyyy'),7);

INSERT INTO client VALUES(ID\_CLIENT.NEXTVAL,'Popescu','Andrei','0213453567',NULL,TO\_DATE('12-4-1998','dd-mm-yyyy'),8);

INSERT INTO client VALUES(ID\_CLIENT.NEXTVAL,'Mirea','Alexandra','0736459294','Dna.',TO\_DATE('13-5-1988','dd-mm-yyyy'),10);

INSERT INTO client VALUES(ID\_CLIENT.NEXTVAL,'Ilarie','David','0762469075','Dl.',TO\_DATE('1-5-1988','dd-mm-yyyy'),9);

INSERT INTO client VALUES(ID\_CLIENT.NEXTVAL,'Sima','Cezar','0216560666','Dl.',TO\_DATE('24-9-1968','dd-mm-yyyy'),6);

INSERT INTO client VALUES(ID\_CLIENT.NEXTVAL,'Petrea','Andreea','0213453567','Dra.',TO\_DATE('30-4-1992','dd-mm-yyyy'),9);

INSERT INTO locatii

VALUES(100,'Bucuresti','Bucuresti');

INSERT INTO locatii

VALUES(200,'Cluj-Napoca','Cluj');

INSERT INTO locatii

VALUES(300,'Timisoara','Timis');

INSERT INTO locatii

VALUES(400,'Oradea','Bihor');

INSERT INTO locatii

VALUES(500,'Iasi','Iasi');

INSERT INTO locatii

VALUES(600,'Alexandria','Teleorman');

CREATE SEQUENCE ID\_CURSA

INCREMENT by 1

START WITH 1

MAXVALUE 1000

NOCYCLE;

INSERT INTO cursa

VALUES(ID\_CURSA.NEXTVAL,'B 24 TAX',101,101,'Bulevardul Unirii 5','Calea Victoriei 118',100);

INSERT INTO cursa

VALUES(ID\_CURSA.NEXTVAL,'B 124 PEL',102,101,'Bulevardul Dacia 141','Bulevardul Libertatii 16',100);

INSERT INTO cursa

VALUES(ID\_CURSA.NEXTVAL,'B 24 TAX',101,102,'Aleea Privighetorilor 65','Strada Polona 45',100);

INSERT INTO cursa

VALUES(ID\_CURSA.NEXTVAL,'IF 745 RBE',106,105,'Aleea Eprubetei 23','Strada Fizicienilor 55',100);

INSERT INTO cursa

VALUES(ID\_CURSA.NEXTVAL,'B 24 TAX',101,103,'Strada Toamnei 2','Strada Beirut 15',100);

INSERT INTO cursa

VALUES(ID\_CURSA.NEXTVAL,'IF 745 RBE',106,104,'Strada Caraiman 13','Calea Motilor 118',200);

INSERT INTO cursa

VALUES(ID\_CURSA.NEXTVAL,'IF 745 RBE',106,104,'Strada Spartacus 3','Strada Tudor Vladimirescu 111',400);

INSERT INTO cursa

VALUES(ID\_CURSA.NEXTVAL,'B 124 PEL',102,105,'Strada Daliei 22','Strada Zorile 75',300);

INSERT INTO cursa

VALUES(ID\_CURSA.NEXTVAL,'B 24 TAX',101,101,'Bulevardul Dacia 141','Bulevardul Basarabia 45',100);

CREATE SEQUENCE ID\_DET

INCREMENT by 1

START WITH 1

MAXVALUE 1000

NOCYCLE;

INSERT INTO detalii\_cursa

VALUES(ID\_DET.NEXTVAL,TO\_DATE('7-4-2021','dd-mm-yyyy'),8,10);

INSERT INTO detalii\_cursa

VALUES(ID\_DET.NEXTVAL,TO\_DATE('8-4-2021','dd-mm-yyyy'),6,7);

INSERT INTO detalii\_cursa

VALUES(ID\_DET.NEXTVAL,TO\_DATE('10-4-2021','dd-mm-yyyy'),7,9);

INSERT INTO detalii\_cursa

VALUES(ID\_DET.NEXTVAL,TO\_DATE('10-4-2021','dd-mm-yyyy'),9,9);

INSERT INTO detalii\_cursa

VALUES(ID\_DET.NEXTVAL,TO\_DATE('10-4-2021','dd-mm-yyyy'),8,8);

INSERT INTO detalii\_cursa

VALUES(ID\_DET.NEXTVAL,TO\_DATE('21-8-2020','dd-mm-yyyy'),8,8);

INSERT INTO detalii\_cursa

VALUES(ID\_DET.NEXTVAL,TO\_DATE('5-7-2020','dd-mm-yyyy'),9,9);

INSERT INTO detalii\_cursa

VALUES(ID\_DET.NEXTVAL,TO\_DATE('8-1-2020','dd-mm-yyyy'),9,7);

INSERT INTO istoric\_sofer

VALUES(101, 7.66, 3);

INSERT INTO istoric\_sofer

VALUES(102, 8, 1);

INSERT INTO istoric\_sofer

VALUES(106, 7, 1);

INSERT INTO istoric\_sofer

VALUES(107, 10, 0);

INSERT INTO discount

VALUES(10,10);

INSERT INTO discount

VALUES(9,7);

INSERT INTO discount

VALUES(8,5);

INSERT INTO discount

VALUES(7,2);

INSERT INTO discount

VALUES(6,1);

INSERT INTO discount

VALUES(5,0);

INSERT INTO discount

VALUES(4,0);

INSERT INTO discount

VALUES(3,0);

INSERT INTO discount

VALUES(2,0);

INSERT INTO discount

VALUES(1,0);

INSERT INTO discount

VALUES(0,0);

CREATE SEQUENCE ID\_FACT

INCREMENT by 1

START WITH 1

MAXVALUE 1000

NOCYCLE;

INSERT INTO factura

VALUES(ID\_FACT.NEXTVAL,105,2,34);

INSERT INTO factura

VALUES(ID\_FACT.NEXTVAL,103,3,51.3);

INSERT INTO factura

VALUES(ID\_FACT.NEXTVAL,104,4,21.1);

INSERT INTO factura

VALUES(ID\_FACT.NEXTVAL,103,5,12.1);

INSERT INTO factura

VALUES(ID\_FACT.NEXTVAL,103,6,60);

INSERT INTO factura

VALUES(ID\_FACT.NEXTVAL,103,7,20);

INSERT INTO factura

VALUES(ID\_FACT.NEXTVAL,105,8,40);

INSERT INTO factura

VALUES(ID\_FACT.NEXTVAL,104,9,60);

INSERT INTO lucreaza\_in

VALUES(101,100);

INSERT INTO lucreaza\_in

VALUES(102,100);

INSERT INTO lucreaza\_in

VALUES(107,300);

INSERT INTO lucreaza\_in

VALUES(106,100);

INSERT INTO lucreaza\_in

VALUES(101,200);

INSERT INTO lucreaza\_in

VALUES(106,200);

INSERT INTO lucreaza\_in

VALUES(106,400);

INSERT INTO lucreaza\_in

VALUES(107,400);

INSERT INTO lucreaza\_in

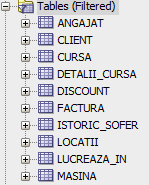
VALUES(107,500);

INSERT INTO lucreaza\_in

VALUES(106,500);

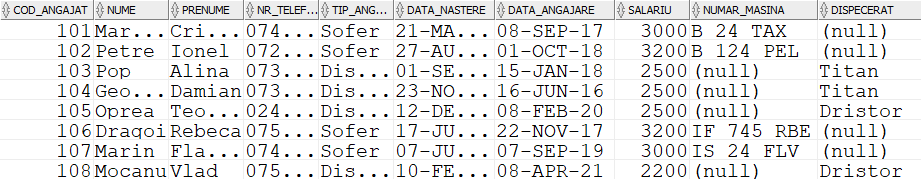
## Rezultatele rularii codului

Am rulat codul pentru a crea tabelele si a insera informatiile in acestea, iar dupa rulare, tabelele arata astfel:

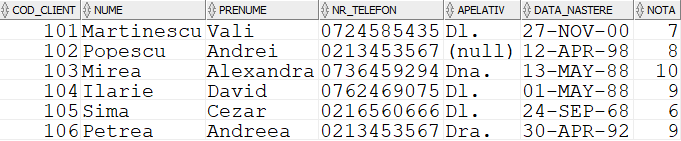


Datele din tabele sunt urmatoarele:

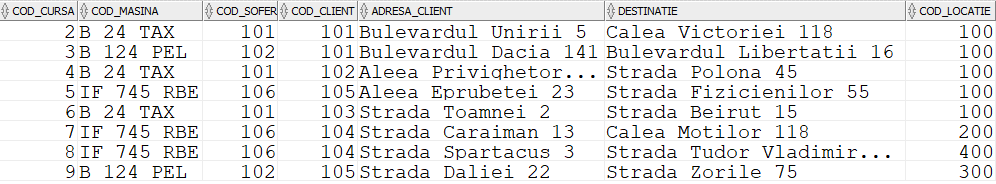
Tabelul ANGAJAT



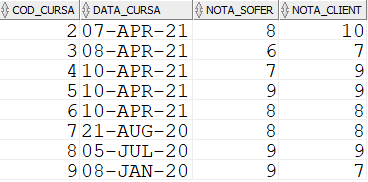
Tabelul CLIENT



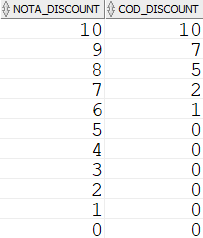
Tabelul CURSA



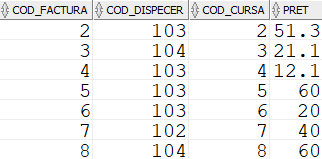
Tabelul DETALII\_CURSA



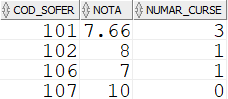
Tabelul DISCOUNT



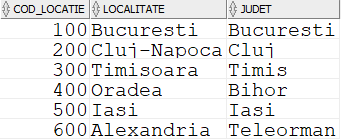
Tabelul FACTURA



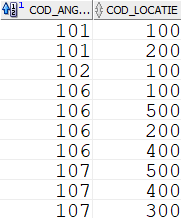
Tabelul ISTORIC\_SOFER



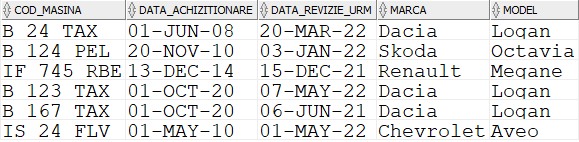
Tabelul LOCATII



Tabelul LUCREAZA\_IN



Tabelul MASINA



# Normalizarea FN1-FN3

## Tabelul ANGAJAT

### Forma normala 1 (FN1)

O relaţie este în prima formă normală dacă fiecărui atribut care o compune îi corespunde o valoare indivizibilă atomică ).

ANGAJAT NON-FN1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cod\_angajat# | Nume | Tip\_angajat | Cod\_locatie# | Oras |
| 100 | Marinescu | Sofer | 200,300 | Bucuresti, Cluj-Napoca |
| 101 | Petre | Sofer | 200,400 | Bucuresti, Timisoara |
| 102 | Pop | Dispecer | (null) | (null) |
| 103 | Georgescu | Dispecer | (null) | (null) |
| 104 | Oprea | Dispecer | (null) | (null) |
| 105 | Dragoi | Sofer | 200,300,500 | Bucuresti, Cluj-Napoca, Oradea |

ANGAJAT FN1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cod\_angajat# | Nume | Tip\_angajat | Cod\_locatie# | Oras |
| 100 | Marinescu | Sofer | 200 | Bucuresti |
| 100 | Marinescu | Sofer | 300 | Cluj-Napoca |
| 101 | Petre | Sofer | 200 | Bucuresti |
| 101 | Petre | Sofer | 400 | Timisoara |
| 102 | Pop | Dispecer | (null) | (null) |
| 103 | Georgescu | Dispecer | (null) | (null) |
| 104 | Oprea | Dispecer | (null) | (null) |
| 105 | Dragoi | Sofer | 200 | Bucuresti |
| 105 | Dragoi | Sofer | 300 | Cluj-Napoca |
| 105 | Dragoi | Sofer | 500 | Oradea |

Locație reprezentând zonele unde șoferii au licență de taxi, nu unde se află.

### Forma normala 2 (FN2)

O relaţie R este în a doua formă normală dacă şi numai dacă:

* relaţia R este în FN1
* fiecare atribut care nu este cheie (nu participă la cheia primară ) este dependent de întreaga cheie primară

TABELUL ANGAJAT FN1, DAR NON FN2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cod\_angajat# | Nume | Tip\_angajat | Cod\_locatie# | Oras |
| 100 | Marinescu | Sofer | 200 | Bucuresti |
| 100 | Marinescu | Sofer | 300 | Cluj-Napoca |
| 101 | Petre | Sofer | 200 | Bucuresti |
| 101 | Petre | Sofer | 400 | Timisoara |
| 102 | Pop | Dispecer | (null) | (null) |
| 103 | Georgescu | Dispecer | (null) | (null) |
| 104 | Oprea | Dispecer | (null) | (null) |
| 105 | Dragoi | Sofer | 200 | Bucuresti |
| 105 | Dragoi | Sofer | 300 | Cluj-Napoca |
| 105 | Dragoi | Sofer | 500 | Oradea |

Astfel avem dependentele pentru relatia ANGAJAT:

* {cod\_angajat#}→{Nume,Tip\_angajat}
* {cod\_locatie#}→{Oras}

Se aplica regula Casey-Delobel:

ANGAJAT (cod\_angajat#, Nume, Tip\_angajat)

LOCATIE\_ANGAJAT (cod\_locatie#, cod\_angajat#, oras)

ANGAJAT

|  |  |  |
| --- | --- | --- |
| Cod\_angajat | Nume | Tip\_angajat |
| 100 | Marinescu | Sofer |
| 101 | Petre | Sofer |
| 102 | Pop | Dispecer |
| 103 | Georgescu | Dispecer |
| 104 | Oprea | Dispecer |
| 105 | Dragoi | Sofer |

LOCATIE\_ANGAJAT

|  |  |  |
| --- | --- | --- |
| Cod\_locatie | Cod\_angajat | Oras |
| 200 | 100 | Bucuresti |
| 300 | 100 | Cluj-Napoca |
| 200 | 101 | Bucuresti |
| 400 | 101 | Timisoara |
| 200 | 105 | Bucuresti |
| 300 | 105 | Cluj-Napoca |
| 500 | 105 | Oradea |

### Forma normala 3 (FN3)

Intuitiv, o relaţie R este în a treia formă normală dacă şi numai dacă:

* relaţia R este în FN2
* fiecare atribut care nu este cheie (nu participă la o cheie) depinde direct de cheia primară

Astfel avem dependentele pentru relatia LOCATIE\_ANGAJAT:

* {cod\_angajat#}→{ }
* {cod\_locatie#}→{Oras}

Se aplică regula Casey Delobel . Relaţia se descompune, prin eliminarea dependenţelor funcţionale tranzitive , în proiecţiile:

ANGAJAT (cod\_angajat#, Nume, Tip\_angajat)

LOCATIE\_ANGAJAT (cod\_locatie#, cod\_angajat#)

LOCATIE(cod\_locatie#, oras)

ANGAJAT

|  |  |  |
| --- | --- | --- |
| Cod\_angajat# | Nume | Tip\_angajat |
| 100 | Marinescu | Sofer |
| 101 | Petre | Sofer |
| 102 | Pop | Dispecer |
| 103 | Georgescu | Dispecer |
| 104 | Oprea | Dispecer |
| 105 | Dragoi | Sofer |

LOCATIE\_ANGAJAT

|  |  |
| --- | --- |
| Cod\_locatie# | Cod\_angajat# |
| 200 | 100 |
| 300 | 100 |
| 200 | 101 |
| 400 | 101 |
| 200 | 105 |
| 300 | 105 |
| 500 | 105 |

LOCATII

|  |  |
| --- | --- |
| Cod\_locatie# | Oras |
| 200 | Bucuresti |
| 300 | Cluj-Napoca |
| 400 | Timisoara |
| 500 | Oradea |

## Tabelul FACTURA

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cod\_factura# | Cod\_dispecer | Cod\_cursa | Pret\_init | Pret\_final |
| 1 | 102 | 1 | 34 | 31.62 |
| 2 | 103 | 2 | 51.3 | 47.71 |
| 3 | 104 | 3 | 21.1 | 20.05 |
| 4 | 103 | 4 | 12 | 11.88 |
| 5 | 103 | 5 | 60 | 55.8 |
| 6 | 103 | 6 | 20 | 19.6 |
| 7 | 102 | 7 | 40 | 39.2 |
| 8 | 104 | 8 | 60 | 59.4 |

Este în FN1 și FN2, respectând condițiile necesare, însă nu este în FN3, atributul pret\_final depinzând de atributul pret\_init, nu de cheia primară. Astfel aveam:

* {cod\_factura#}→{cod\_dispecer, cod\_cursa, pret\_init}
* {cod\_factura#}→{pret\_init}→{pret\_final}

Pentru a aduce relația FACTURA în FN3, folosim un singur atribut, cu numele preț, astfel că acesta nu depinde decât de cheia primară.

Pentru a modifica prețul în funcție de ce discount primește clientul, se va folosi o cerere UPDATE după introducerea datelor în tabel.

|  |  |  |  |
| --- | --- | --- | --- |
| Cod\_factura# | Cod\_dispecer | Cod\_cursa | Pret |
| 1 | 102 | 1 | 34 |
| 2 | 103 | 2 | 51.3 |
| 3 | 104 | 3 | 21.1 |
| 4 | 103 | 4 | 12.7 |
| 5 | 103 | 5 | 60 |
| 6 | 103 | 6 | 25.7 |
| 7 | 102 | 7 | 44.2 |
| 8 | 104 | 8 | 45.9 |

Celelalte tabele sunt în FN1, FN2 și FN3, respectând condițiile necesare.

# Cerintele 6 – 13

## Cerinta 6

Creati o procedura care, la executie, afiseaza toti angajatii si, daca anagajatul este sofer, afiseaza orasele in care poate conduce:

CREATE OR REPLACE PROCEDURE ex6

IS

TYPE tablou\_imbricat IS TABLE OF NUMBER;

TYPE tablou\_indexat IS TABLE OF NUMBER INDEX BY PLS\_INTEGER;

coduri\_locatii tablou\_imbricat := tablou\_imbricat();

coduri\_angajati tablou\_indexat;

cod\_ang NUMBER(4);

nr\_ang NUMBER(3);

nume angajat.nume%type;

prenume angajat.prenume%type;

oras locatii.localitate%type;

rol\_ang angajat.tip\_angajat%type;

BEGIN

SELECT count(\*) into nr\_ang from angajat;

SELECT cod\_angajat bulk collect into coduri\_angajati from angajat;

for a in coduri\_angajati.FIRST..coduri\_angajati.LAST LOOP

SELECT nume,prenume,tip\_angajat INTO nume,prenume,rol\_ang FROM angajat WHERE cod\_angajat=coduri\_angajati(a);

cod\_ang:=coduri\_angajati(a);

SELECT cod\_locatie bulk collect into coduri\_locatii from lucreaza\_in where cod\_angajat=cod\_ang;

DBMS\_OUTPUT.PUT\_LINE('Angajatul '||nume||' '||prenume||' cu codul '||cod\_ang||':');

if rol\_ang='Sofer' then

if coduri\_locatii.count>0 then

DBMS\_OUTPUT.PUT\_LINE('Lucreaza in orasele:');

FOR i IN coduri\_locatii.FIRST..coduri\_locatii.LAST LOOP

SELECT localitate INTO oras FROM locatii where cod\_locatie=coduri\_locatii(i);

DBMS\_OUTPUT.PUT\_LINE(oras);

END LOOP;

else DBMS\_OUTPUT.PUT\_LINE('Nu poate lucra in nici un oras');

end if;

else DBMS\_OUTPUT.PUT\_LINE('Nu este sofer');

end if;

DBMS\_OUTPUT.NEW\_LINE;

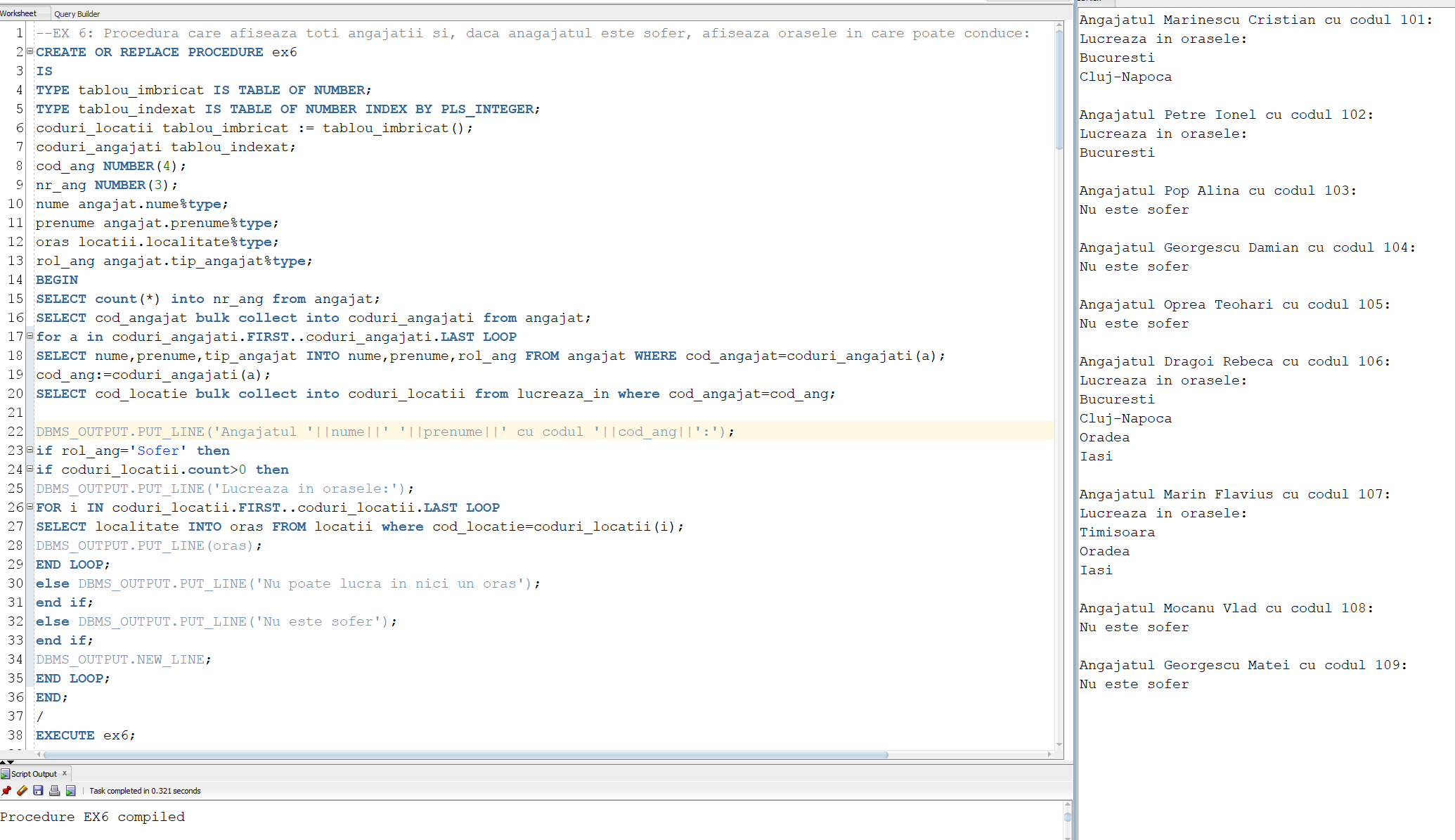
END LOOP;

END;

/

EXECUTE ex6;

Executia cerintei 6:



## Cerinta 7

Creati o procedura care, la executie, afiseaza toate masinile, soferul lor si nr de curse facute (daca au):

CREATE OR REPLACE PROCEDURE ex7 IS

nr\_curse NUMBER(4);

cursor masina\_cursor is

select cod\_masina, marca, model

from masina;

CURSOR angajat\_cursor ( cod\_masina masina.cod\_masina%type) IS

select nume, prenume, salariu

from angajat

where numar\_masina = cod\_masina;

BEGIN

for m in masina\_cursor loop

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

DBMS\_OUTPUT.PUT\_LINE ('Masina '||m.cod\_masina||' '||m.marca||' '||m.model);

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

SELECT count(\*) into nr\_curse from cursa where cod\_masina=m.cod\_masina;

if nr\_curse =0 then

DBMS\_OUTPUT.PUT\_LINE( 'Aceasta masina nu a facut curse');

else for a in angajat\_cursor(m.cod\_masina) loop

DBMS\_OUTPUT.PUT\_LINE('Sofer: ' || a.nume || ' ' || a.prenume || ', Numar curse: ' || nr\_curse);

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

end loop;

end if;

DBMS\_OUTPUT.NEW\_LINE;

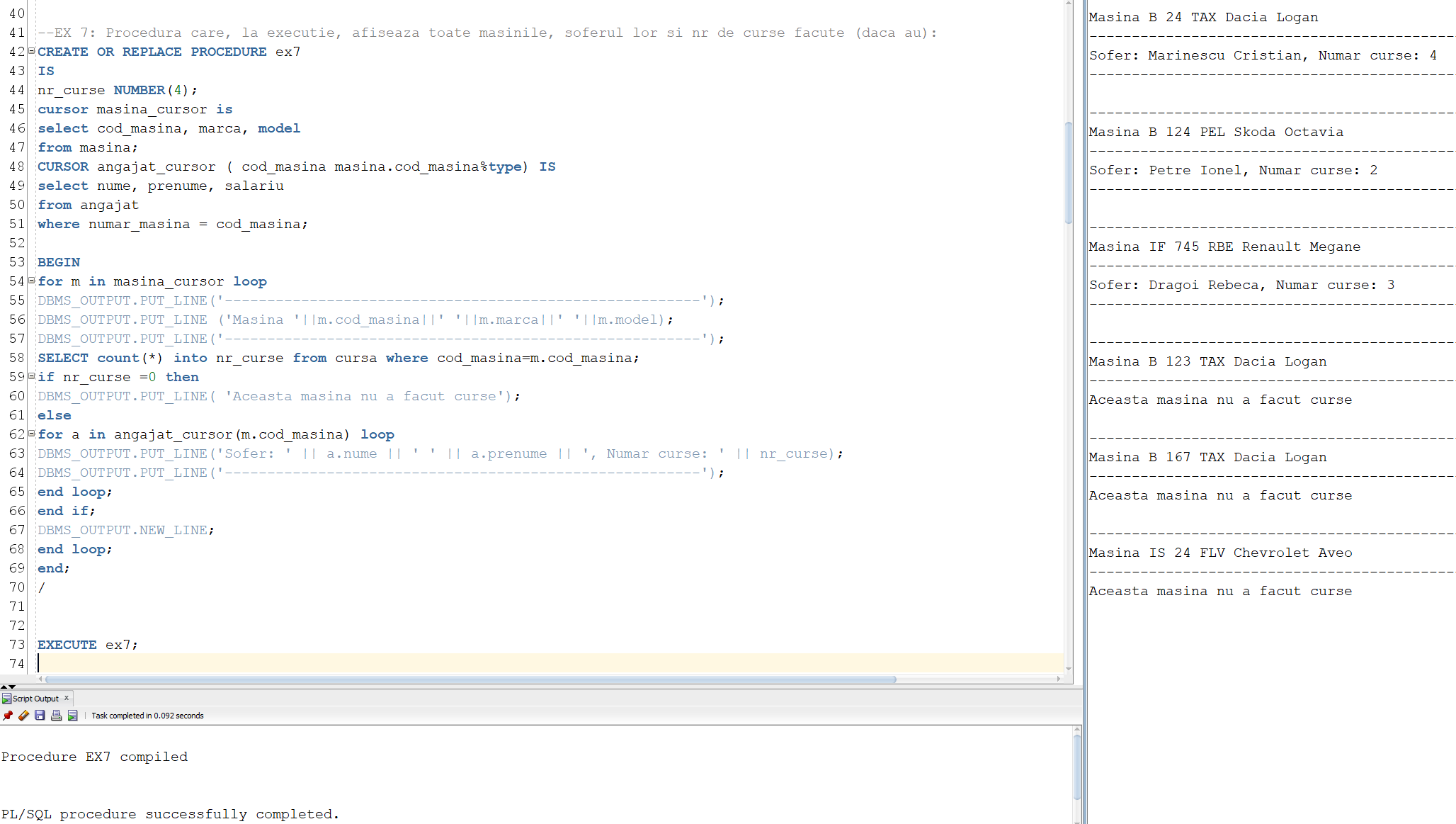
end loop;

end;

/

EXECUTE ex7;

Executia cerintei 7:



## Cerinta 8

Creati o functie care primeste ca parametru numele angajatului si afiseaza detalii despre acesta si in functie de tipul de angajat, numarul masinii si cursele sau dispeceratul si numarul de facturi emise:

CREATE OR REPLACE FUNCTION ex8

(nume\_ang IN angajat.nume%type)

RETURN VARCHAR2

IS

rezultat VARCHAR2(200);

cod\_ang angajat.cod\_angajat%type;

prenume angajat.prenume%type;

tip\_ang angajat.tip\_angajat%type;

sal angajat.salariu%type;

nr\_masina angajat.numar\_masina%type;

dispecerat angajat.dispecerat%type;

nr NUMBER(4);

oras locatii.localitate%type;

nr\_telefon angajat.nr\_telefon%type;

nota istoric\_sofer.nota%type;

marca masina.marca%type;

model masina.model%type;

BEGIN

BEGIN

select cod\_angajat, prenume, tip\_angajat, salariu, numar\_masina, dispecerat

into cod\_ang,prenume,tip\_ang,sal,nr\_masina,dispecerat

from angajat

where upper(nume\_ang) = upper(nume);

EXCEPTION

WHEN NO\_DATA\_FOUND

THEN

DBMS\_OUTPUT.PUT\_LINE('Angajatul nu exista');

RAISE\_APPLICATION\_ERROR (-20000,'Angajatul nu exista');

END;

rezultat:=rezultat||nume\_ang||' '||prenume||', '||tip\_ang||', salariu '||CAST(sal as VARCHAR2);

if tip\_ang='Sofer' then

SELECT count(\*) INTO nr FROM cursa where cod\_sofer=cod\_ang group by cod\_sofer;

SELECT a.nr\_telefon, i.nota, m.marca, m.model into nr\_telefon, nota, marca, model

from angajat a join istoric\_sofer i on a.cod\_angajat=i.cod\_sofer join masina m on a.numar\_masina=m.cod\_masina

where cod\_ang=a.cod\_angajat;

rezultat:=rezultat||', numar masina '||nr\_masina||', nr curse '||CAST(nr as VARCHAR2)||', nr telefon '||nr\_telefon

||', nota sofer '||nota||', masina '||marca||' '||model;

elsif tip\_ang='Dispecer' then

SELECT count(\*) INTO nr FROM factura where cod\_dispecer=cod\_ang group by cod\_dispecer;

rezultat:=rezultat||', dispecerat '||dispecerat||', nr facturi '||CAST(nr as VARCHAR2);

else RETURN 'Angajatul nu este sofer sau dispecer';

end if;

RETURN rezultat;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Angajatul nu are curse realizate sau facturi emise');

RAISE\_APPLICATION\_ERROR (-20001,'Angajatul nu are curse realizate sau facturi emise');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Mai multi angajati cu acelasi nume');

RAISE\_APPLICATION\_ERROR (-20002, 'Mai multi angajati cu acelasi nume');

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE(ex8('Pop'));

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE(ex8('Marinescu'));

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE(ex8('Georgescu'));

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE(ex8('Marin'));

END;

/

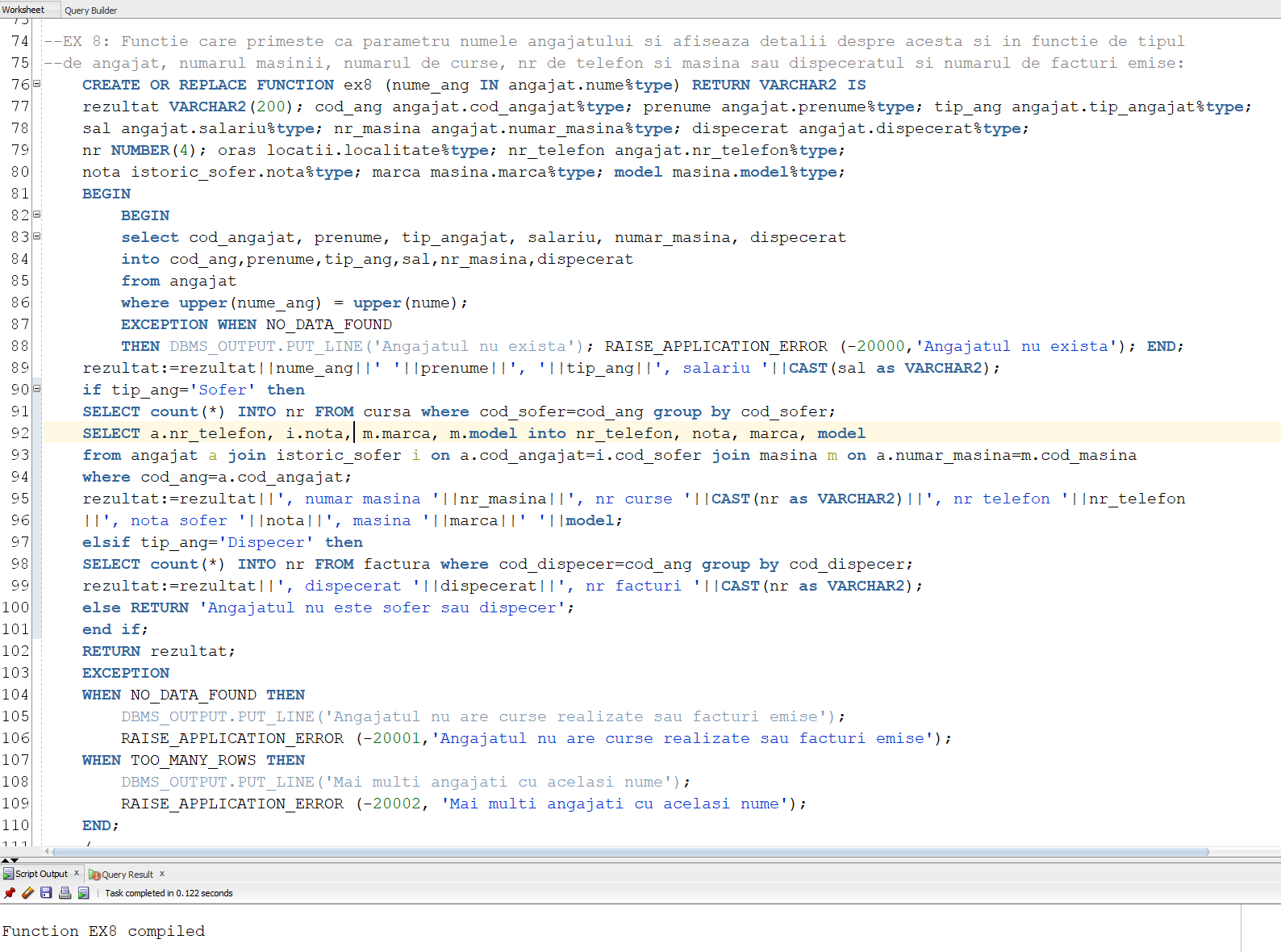
BEGIN

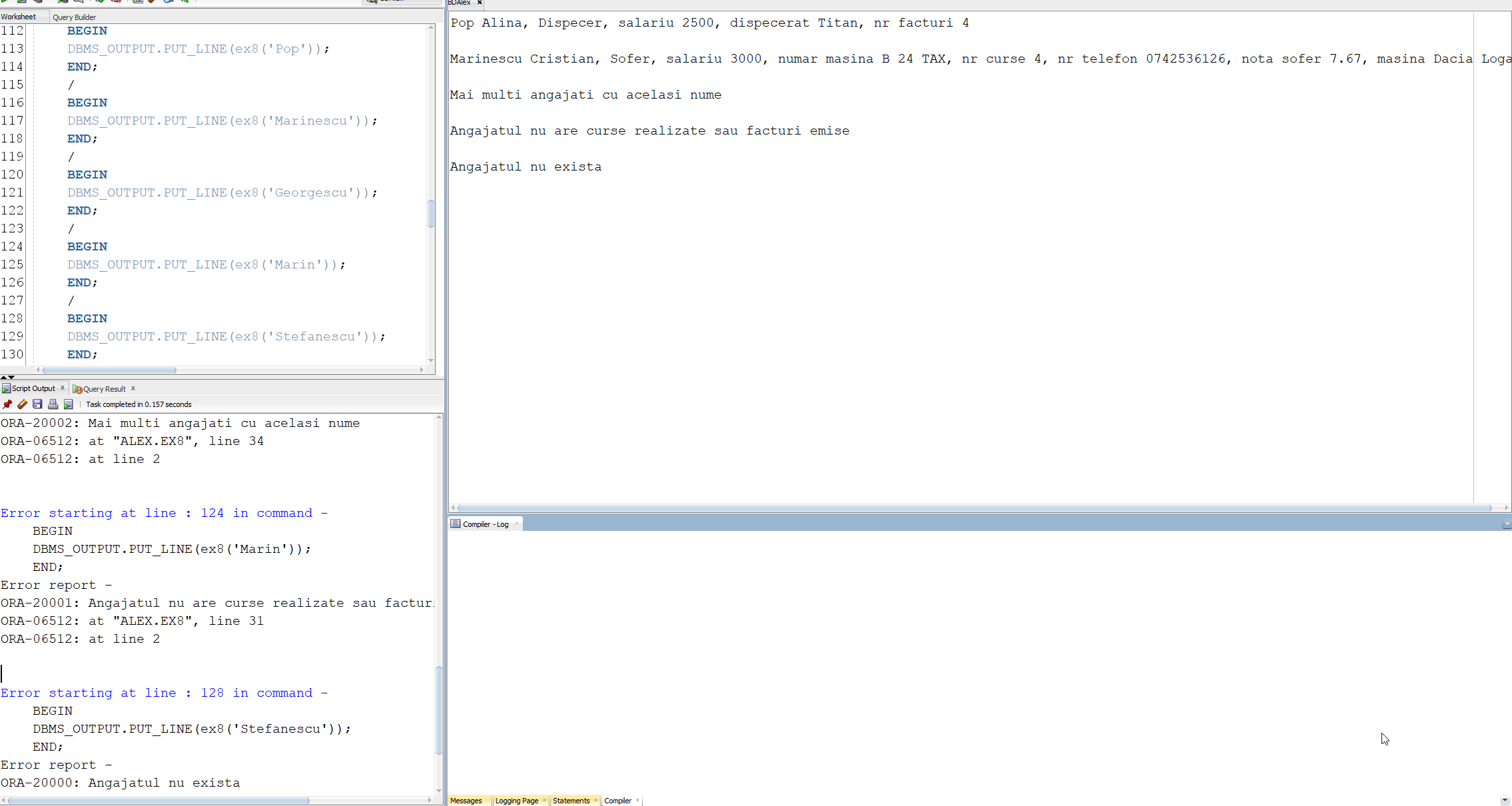
DBMS\_OUTPUT.PUT\_LINE(ex8('Stefanescu'));

END;

/

Executia cerintei 8:





## Cerinta 9

Creati o procedura care primeste ca parametru un nume de strada si afiseaza detalii din mai multe tabele despre cursa, masina sau sofer:

CREATE OR REPLACE PROCEDURE ex9

(adresa\_c IN cursa.adresa\_client%type)

AS

id\_cursa cursa.cod\_cursa%type;

cod\_sofer cursa.cod\_sofer%type;

v\_adresa cursa.adresa\_client%type;

destinatie cursa.adresa\_client%type;

cod\_loc cursa.cod\_locatie%type;

pret factura.pret%type;

v\_masina masina.cod\_masina%type;

marca masina.marca%type;

v\_model masina.model%type;

nota\_sofer detalii\_cursa.nota\_sofer%type;

localit locatii.localitate%type;

nume angajat.nume%type;

prenume angajat.prenume%type;

BEGIN

SELECT cod\_cursa into id\_cursa

from cursa where UPPER(adresa\_client) LIKE '%'||UPPER(adresa\_c)||'%';

SELECT c.cod\_sofer,c.adresa\_client,c.destinatie,c.cod\_locatie,f.pret,a.numar\_masina,a.nume,a.prenume,

m.marca,m.model, d.nota\_sofer, l.localitate into cod\_sofer,v\_adresa,destinatie,cod\_loc,pret,v\_masina,nume,prenume,

marca,v\_model,nota\_sofer,localit from cursa c join factura f on c.cod\_cursa=f.cod\_cursa join angajat a on

c.cod\_sofer=a.cod\_angajat join masina m on a.numar\_masina=m.cod\_masina join detalii\_cursa d on d.cod\_cursa=c.cod\_cursa

join locatii l on l.cod\_locatie=c.cod\_locatie

where c.cod\_cursa=id\_cursa;

DBMS\_OUTPUT.PUT\_LINE('Cursa ' || id\_cursa || ' din orasul '|| localit || ' de la adresa '|| v\_adresa ||' la adresa '||

destinatie || chr(10)||' a fost facuta cu masina '|| marca ||' '|| v\_model ||', soferul a primit nota '|| nota\_sofer

|| ' iar pretul cursei a fost '|| pret);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Cursa care porneste de la adresa specificata nu exista');

RAISE\_APPLICATION\_ERROR (-20000,'Cursa care porneste de la adresa specificata nu exista');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Mai multe curse cu aceeasi locatie, dati o locatie mai specifica');

RAISE\_APPLICATION\_ERROR (-20001, 'Mai multe curse cu aceeasi locatie, dati o locatie mai specifica');

END;

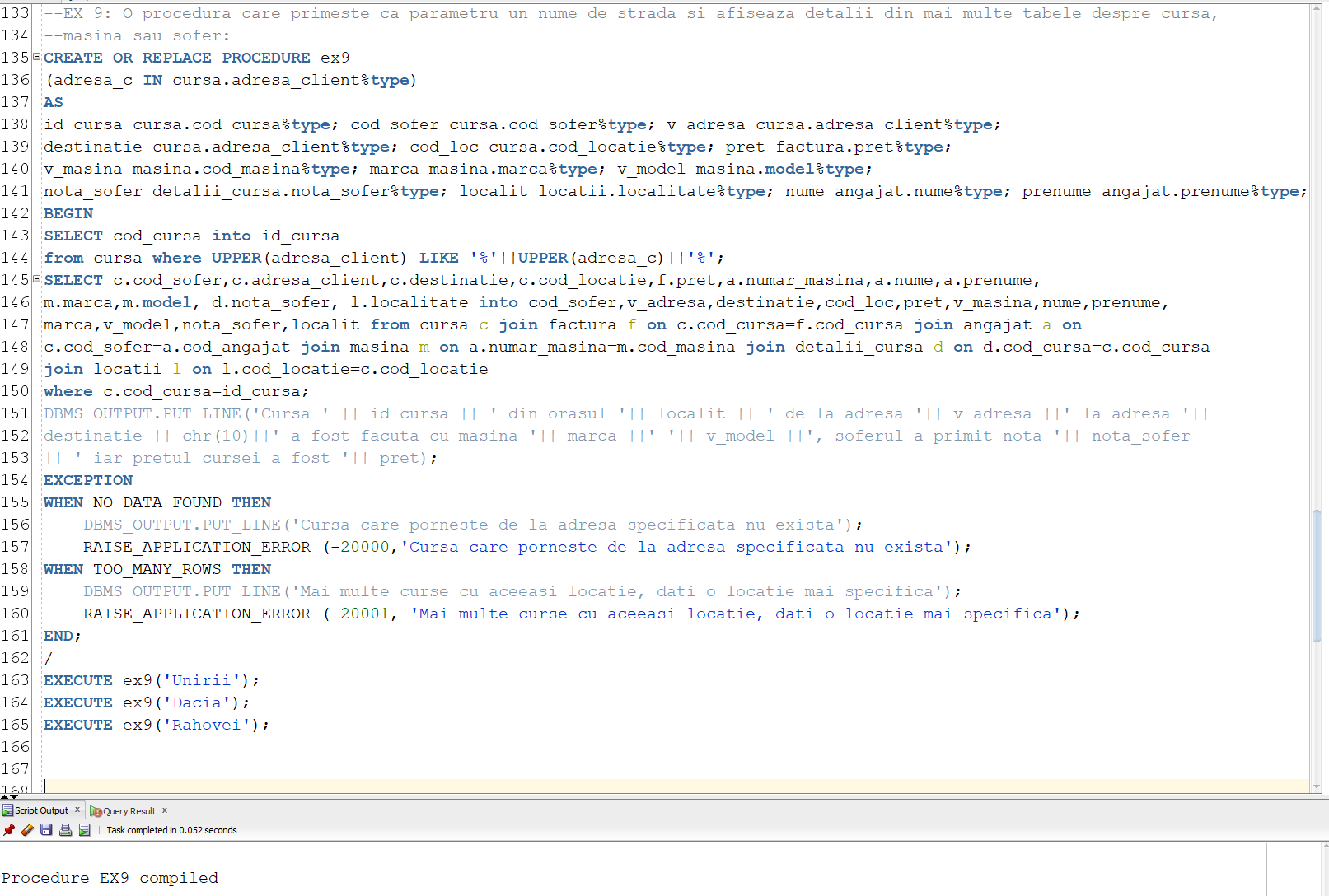
/

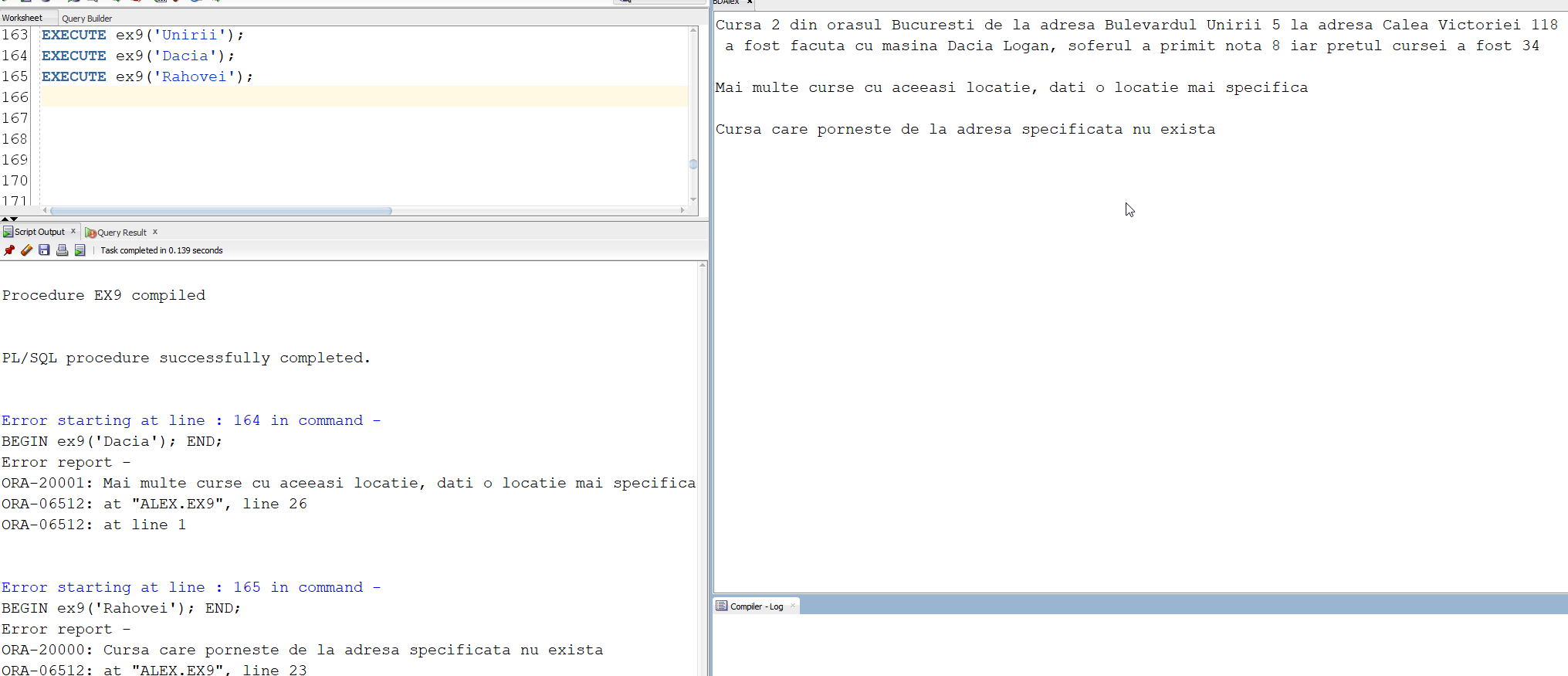
EXECUTE ex9('Unirii');

EXECUTE ex9('Dacia');

EXECUTE ex9('Rahovei');

Executie cerinta 9:





## Cerinta 10

Creati un trigger pentru actualizare automata a notei soferului si a clientului dupa o cursa:

CREATE OR REPLACE TRIGGER trig\_ex10

AFTER INSERT OR UPDATE on detalii\_cursa

DECLARE

cursor client\_cursor is

SELECT c.cod\_client cod,avg(d.nota\_client) nota

from detalii\_cursa d join cursa c on d.cod\_cursa=c.cod\_cursa

group by c.cod\_client;

cursor soferi\_cursor is

SELECT c.cod\_sofer cod,avg(d.nota\_sofer) nota

from detalii\_cursa d join cursa c on d.cod\_cursa=c.cod\_cursa

group by c.cod\_sofer;

nr\_curse istoric\_sofer.numar\_curse%type;

BEGIN

for s in soferi\_cursor loop

SELECT count(\*) into nr\_curse from cursa where cod\_sofer=s.cod;

UPDATE istoric\_sofer set nota=s.nota,numar\_curse=nr\_curse where cod\_sofer=s.cod;

end loop;

for c in client\_cursor loop

UPDATE client set nota=c.nota where cod\_client=c.cod;

end loop;

END;

/

UPDATE detalii\_cursa set nota\_sofer=8,nota\_client=8 where cod\_cursa=7;

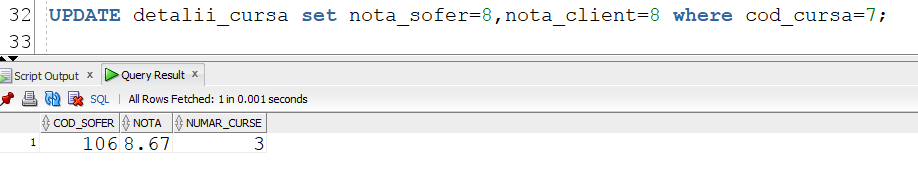
SELECT \* FROM istoric\_sofer;

SELECT \* FROM client;

UPDATE detalii\_cursa set nota\_sofer=5,nota\_client=5 where cod\_cursa=7;

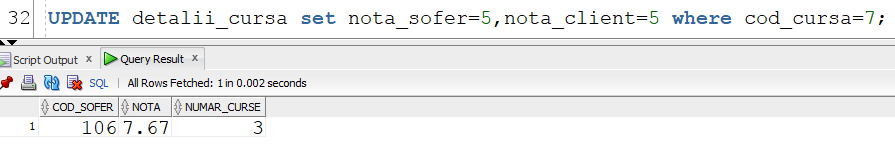
Testare cerinta 10:

Initial, cursa cu codul 7 are nota soferului 8 si nota clientului 8, astfel ca nota medie a soferului este 8.67 si nota medie a clientului este 9.





Dupa actualizarea detaliilor cursei cu codul 7, unde ambele note devin 5, in loc de 8, nota medie a soferului devine 7.67, de la 8.67, iar nota medie a clientului devine 7 de la 9.





## Cerinta 11

Creati un trigger care impune ca notele trebuie sa fie intre 1 si 10:

CREATE OR REPLACE TRIGGER trig\_ex11

BEFORE UPDATE OF nota\_sofer,nota\_client ON detalii\_cursa

FOR EACH ROW

WHEN ((NEW.nota\_sofer>10 or NEW.nota\_sofer<1) or (NEW.nota\_client>10 or NEW.nota\_client<1))

BEGIN

RAISE\_APPLICATION\_ERROR (-20000, 'Nota invalida!');

END;

/

UPDATE detalii\_cursa set nota\_sofer=8,nota\_client=5 where cod\_cursa=7;

--la toate de mai jos da eroare

UPDATE detalii\_cursa set nota\_sofer=18,nota\_client=-1 where cod\_cursa=7;

UPDATE detalii\_cursa set nota\_sofer=8,nota\_client=-1 where cod\_cursa=7;

UPDATE detalii\_cursa set nota\_sofer=-8,nota\_client=5 where cod\_cursa=7;

UPDATE detalii\_cursa set nota\_sofer=-8,nota\_client=15 where cod\_cursa=7;

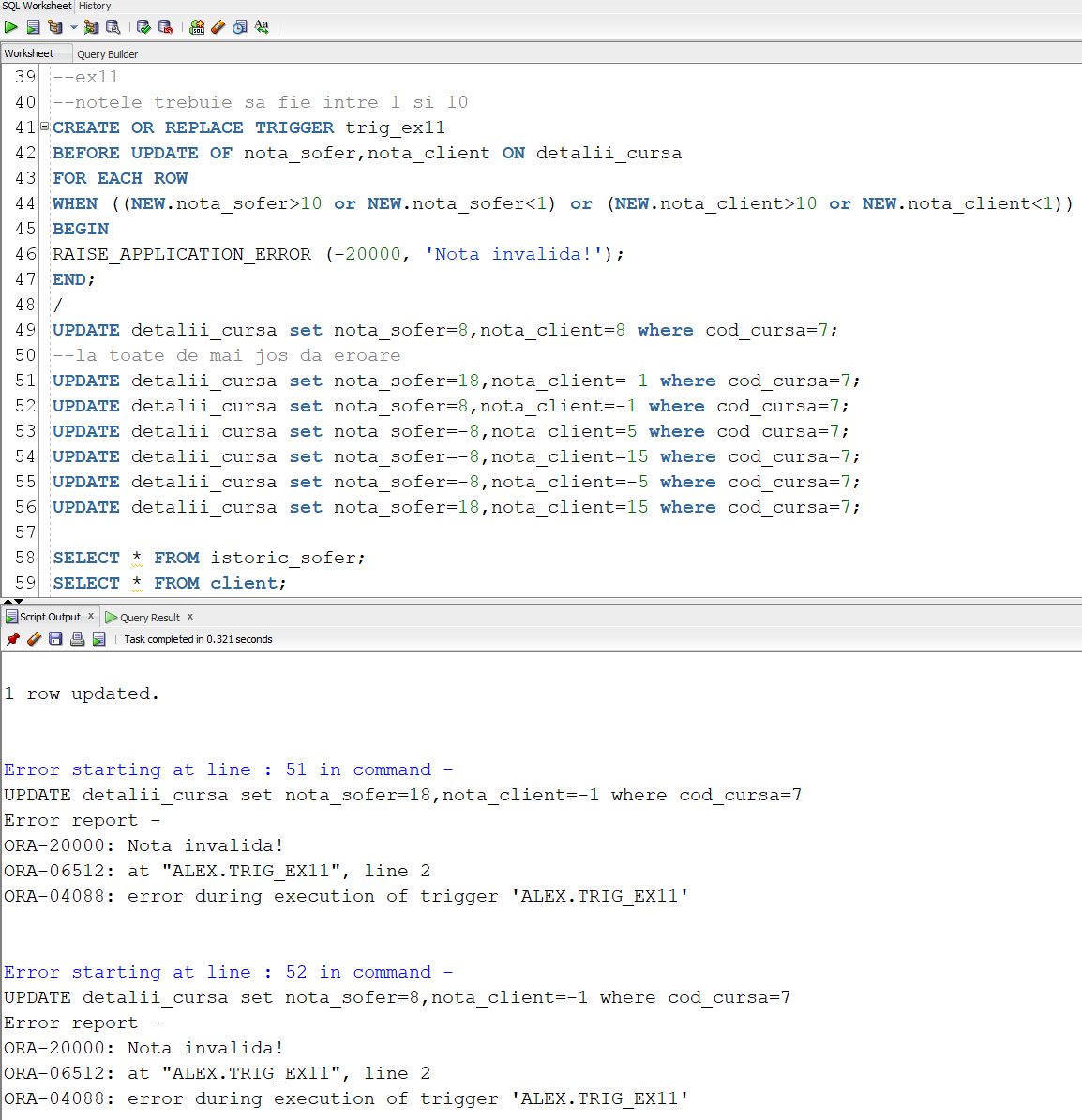
UPDATE detalii\_cursa set nota\_sofer=-8,nota\_client=-5 where cod\_cursa=7;

UPDATE detalii\_cursa set nota\_sofer=18,nota\_client=15 where cod\_cursa=7;

SELECT \* FROM istoric\_sofer;

SELECT \* FROM client;

Testare cerinta 11:



## Cerinta 12

Creati un trigger care interzice drop la tabele:

CREATE OR REPLACE TRIGGER trig\_ex12

AFTER DROP ON SCHEMA

BEGIN

RAISE\_APPLICATION\_ERROR (-20000, 'Nu se poate da drop la tabelele existente!');

ROLLBACK;

END;

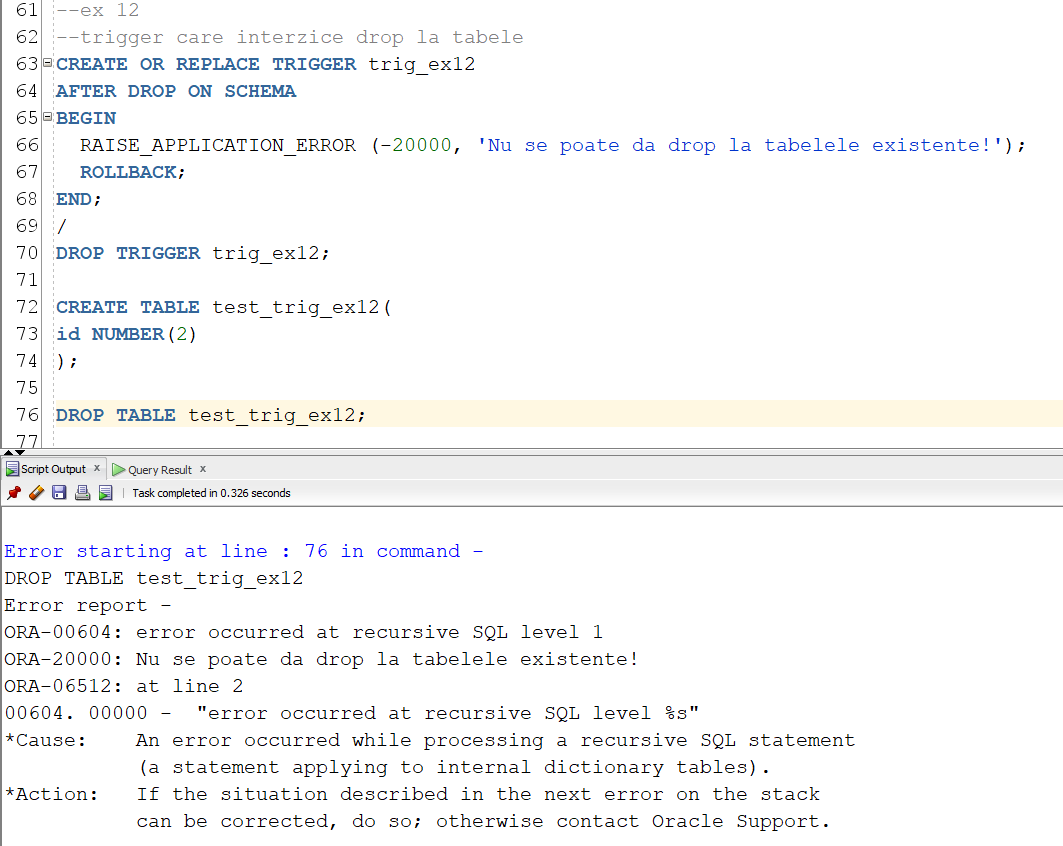
/

CREATE TABLE test\_trig\_ex12(

id NUMBER(2));

DROP TABLE test\_trig\_ex12;

Testare cerinta 12:



## Cerinta 13

CREATE OR REPLACE PACKAGE pachet1 is

PROCEDURE ex6;

PROCEDURE ex7;

FUNCTION ex8(nume\_ang IN angajat.nume%type)

RETURN VARCHAR2;

PROCEDURE ex9(adresa\_c IN cursa.adresa\_client%type);

END pachet1;

/

CREATE OR REPLACE PACKAGE BODY pachet1 is

PROCEDURE ex6

IS

TYPE tablou\_imbricat IS TABLE OF NUMBER;

TYPE tablou\_indexat IS TABLE OF NUMBER INDEX BY PLS\_INTEGER;

coduri\_locatii tablou\_imbricat := tablou\_imbricat();

coduri\_angajati tablou\_indexat;

cod\_ang NUMBER(4);

nr\_ang NUMBER(3);

nume angajat.nume%type;

prenume angajat.prenume%type;

oras locatii.localitate%type;

rol\_ang angajat.tip\_angajat%type;

BEGIN

SELECT count(\*) into nr\_ang from angajat;

SELECT cod\_angajat bulk collect into coduri\_angajati from angajat;

for a in coduri\_angajati.FIRST..coduri\_angajati.LAST LOOP

SELECT nume,prenume,tip\_angajat INTO nume,prenume,rol\_ang FROM angajat WHERE cod\_angajat=coduri\_angajati(a);

cod\_ang:=coduri\_angajati(a);

SELECT cod\_locatie bulk collect into coduri\_locatii from lucreaza\_in where cod\_angajat=cod\_ang;

DBMS\_OUTPUT.PUT\_LINE('Angajatul '||nume||' '||prenume||' cu codul '||cod\_ang||':');

if rol\_ang='Sofer' then

if coduri\_locatii.count>0 then

DBMS\_OUTPUT.PUT\_LINE('Lucreaza in orasele:');

FOR i IN coduri\_locatii.FIRST..coduri\_locatii.LAST LOOP

SELECT localitate INTO oras FROM locatii where cod\_locatie=coduri\_locatii(i);

DBMS\_OUTPUT.PUT\_LINE(oras);

END LOOP;

else DBMS\_OUTPUT.PUT\_LINE('Nu poate lucra in nici un oras');

end if;

else DBMS\_OUTPUT.PUT\_LINE('Nu este sofer');

end if;

DBMS\_OUTPUT.NEW\_LINE;

END LOOP;

END;

PROCEDURE ex7

IS

nr\_curse NUMBER(4);

cursor masina\_cursor is

select cod\_masina, marca, model

from masina;

CURSOR angajat\_cursor ( cod\_masina masina.cod\_masina%type) IS

select nume, prenume, salariu

from angajat

where numar\_masina = cod\_masina;

BEGIN

for m in masina\_cursor loop

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

DBMS\_OUTPUT.PUT\_LINE ('Masina '||m.cod\_masina||' '||m.marca||' '||m.model);

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

SELECT count(\*) into nr\_curse from cursa where cod\_masina=m.cod\_masina;

if nr\_curse =0 then

DBMS\_OUTPUT.PUT\_LINE( 'Aceasta masina nu a facut curse');

else

for a in angajat\_cursor(m.cod\_masina) loop

DBMS\_OUTPUT.PUT\_LINE('Sofer: ' || a.nume || ' ' || a.prenume || ', Numar curse: ' || nr\_curse);

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

end loop;

end if;

DBMS\_OUTPUT.NEW\_LINE;

end loop;

end;

FUNCTION ex8

(nume\_ang IN angajat.nume%type)

RETURN VARCHAR2

IS

rezultat VARCHAR2(200);

cod\_ang angajat.cod\_angajat%type;

prenume angajat.prenume%type;

tip\_ang angajat.tip\_angajat%type;

sal angajat.salariu%type;

nr\_masina angajat.numar\_masina%type;

dispecerat angajat.dispecerat%type;

nr NUMBER(4);

oras locatii.localitate%type;

nr\_telefon angajat.nr\_telefon%type;

nota istoric\_sofer.nota%type;

marca masina.marca%type;

model masina.model%type;

BEGIN

BEGIN

select cod\_angajat, prenume, tip\_angajat, salariu, numar\_masina, dispecerat

into cod\_ang,prenume,tip\_ang,sal,nr\_masina,dispecerat

from angajat

where upper(nume\_ang) = upper(nume);

EXCEPTION

WHEN NO\_DATA\_FOUND

THEN

DBMS\_OUTPUT.PUT\_LINE('Angajatul nu exista');

RAISE\_APPLICATION\_ERROR (-20000,'Angajatul nu exista');

END;

rezultat:=rezultat||nume\_ang||' '||prenume||', '||tip\_ang||', salariu '||CAST(sal as VARCHAR2);

if tip\_ang='Sofer' then

SELECT count(\*) INTO nr FROM cursa where cod\_sofer=cod\_ang group by cod\_sofer;

SELECT a.nr\_telefon, i.nota, m.marca, m.model into nr\_telefon, nota, marca, model

from angajat a join istoric\_sofer i on a.cod\_angajat=i.cod\_sofer join masina m on a.numar\_masina=m.cod\_masina

where cod\_ang=a.cod\_angajat;

rezultat:=rezultat||', numar masina '||nr\_masina||', nr curse '||CAST(nr as VARCHAR2)||', nr telefon '||nr\_telefon

||', nota sofer '||nota||', masina '||marca||' '||model;

elsif tip\_ang='Dispecer' then

SELECT count(\*) INTO nr FROM factura where cod\_dispecer=cod\_ang group by cod\_dispecer;

rezultat:=rezultat||', dispecerat '||dispecerat||', nr facturi '||CAST(nr as VARCHAR2);

else RETURN 'Angajatul nu este sofer sau dispecer';

end if;

RETURN rezultat;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Angajatul nu are curse realizate sau facturi emise');

RAISE\_APPLICATION\_ERROR (-20001,'Angajatul nu are curse realizate sau facturi emise');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Mai multi angajati cu acelasi nume');

RAISE\_APPLICATION\_ERROR (-20002, 'Mai multi angajati cu acelasi nume');

END;

PROCEDURE ex9

(adresa\_c IN cursa.adresa\_client%type)

AS

id\_cursa cursa.cod\_cursa%type; cod\_sofer cursa.cod\_sofer%type; v\_adresa cursa.adresa\_client%type;

destinatie cursa.adresa\_client%type; cod\_loc cursa.cod\_locatie%type; pret factura.pret%type;

v\_masina masina.cod\_masina%type; marca masina.marca%type; v\_model masina.model%type;

nota\_sofer detalii\_cursa.nota\_sofer%type; localit locatii.localitate%type; nume angajat.nume%type; prenume angajat.prenume%type;

BEGIN

SELECT cod\_cursa into id\_cursa

from cursa where UPPER(adresa\_client) LIKE '%'||UPPER(adresa\_c)||'%';

SELECT c.cod\_sofer,c.adresa\_client,c.destinatie,c.cod\_locatie,f.pret,a.numar\_masina,a.nume,a.prenume,

m.marca,m.model, d.nota\_sofer, l.localitate into cod\_sofer,v\_adresa,destinatie,cod\_loc,pret,v\_masina,nume,prenume,

marca,v\_model,nota\_sofer,localit from cursa c join factura f on c.cod\_cursa=f.cod\_cursa join angajat a on

c.cod\_sofer=a.cod\_angajat join masina m on a.numar\_masina=m.cod\_masina join detalii\_cursa d on d.cod\_cursa=c.cod\_cursa

join locatii l on l.cod\_locatie=c.cod\_locatie

where c.cod\_cursa=id\_cursa;

DBMS\_OUTPUT.PUT\_LINE('Cursa ' || id\_cursa || ' din orasul '|| localit || ' de la adresa '|| v\_adresa ||' la adresa '||

destinatie || chr(10)||' a fost facuta cu masina '|| marca ||' '|| v\_model ||', soferul a primit nota '|| nota\_sofer

|| ' iar pretul cursei a fost '|| pret);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Cursa care porneste de la adresa specificata nu exista');

RAISE\_APPLICATION\_ERROR (-20000,'Cursa care porneste de la adresa specificata nu exista');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Mai multe curse cu aceeasi locatie, dati o locatie mai specifica');

RAISE\_APPLICATION\_ERROR (-20001, 'Mai multe curse cu aceeasi locatie, dati o locatie mai specifica');

END;

END;

Testare exercitiul 13:

