

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
Entități	1
Relații	2
Atribute.....	3
Diagrama entitate – relație – Cerinta 2.....	7
Diagrama conceptuală – Cerinta 3.....	8
Schemele relaționale	9
Diagrama entitate-relatie, cu atribute.....	10
Crearea bazei de date si inserarea informatiilor – Cerintele 4 si 5	11
Codul pentru creare si inserare	11
Rezultatele rularii codului	21
Normalizarea FN1-FN3.....	24
Tabelul ANGAJAT	24
Forma normala 1 (FN1).....	24
Forma normala 2 (FN2).....	25
Forma normala 3 (FN3).....	26
Tabelul FACTURA.....	27
Celelalte tabele.....	28
Cerintele 6 – 13.....	28
Cerinta 6	28
Cerinta 7	30
Cerinta 8	32
Cerinta 9	36
Cerinta 10	39
Cerinta 11	41
Cerinta 12	43
Cerinta 13	44

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, restricțiile pe care trebuie să le respecte și vom încerca să construim diagrama E/R corespunzătoare. Vom considera, în abordarea inițială, anumite situații care nu sunt optime, în sensul că pot genera redundanță, anomalii la reactualizări sau nu permit rezolvarea anumitor interogări asupra modelului. Vom încerca să arătăm care sunt deficiențele modelului, situațiile care le-au generat și cum pot fi corectate (parțial 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 restricții de funcționare.

- 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 entitățile modelului de date, dând o descriere completă a fiecăreia. De asemenea, pentru fiecare entitate se va preciza cheia primară.

Toate entitățile care vor fi prezentate sunt independente, cu excepția subentităților 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_SOFTER = 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_softer.

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.

Relații

Vom prezenta relațiile modelului de date, dând o descriere completă a fiecăreia. De fapt, denumirile acestor legături sunt sugestive, reflectând conținutul acestora și entitățile pe care le leagă. Pentru fiecare relație se va preciza cardinalitatea minimă și maximă.

SOFTER_face_CURSA = relație care leagă entitățile SOFTER 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).

SOFTER_are_ISTORIC_SOFTER = relație care leagă entitățile SOFTER și ISTORIC_SOFTER, 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_SOFTER = relație care leagă entitățile VEHICUL și SOFTER, 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 minimă 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 nașterii 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 nașterii angajatului respectiv.

Subentitatea SOFER are ca atribut:

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 atribut:

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 atribut:

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 attribute:

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 attribute:

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 attribute:

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 attribute:

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 attribute:

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 attribute:

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 – relație – Cerinta 2

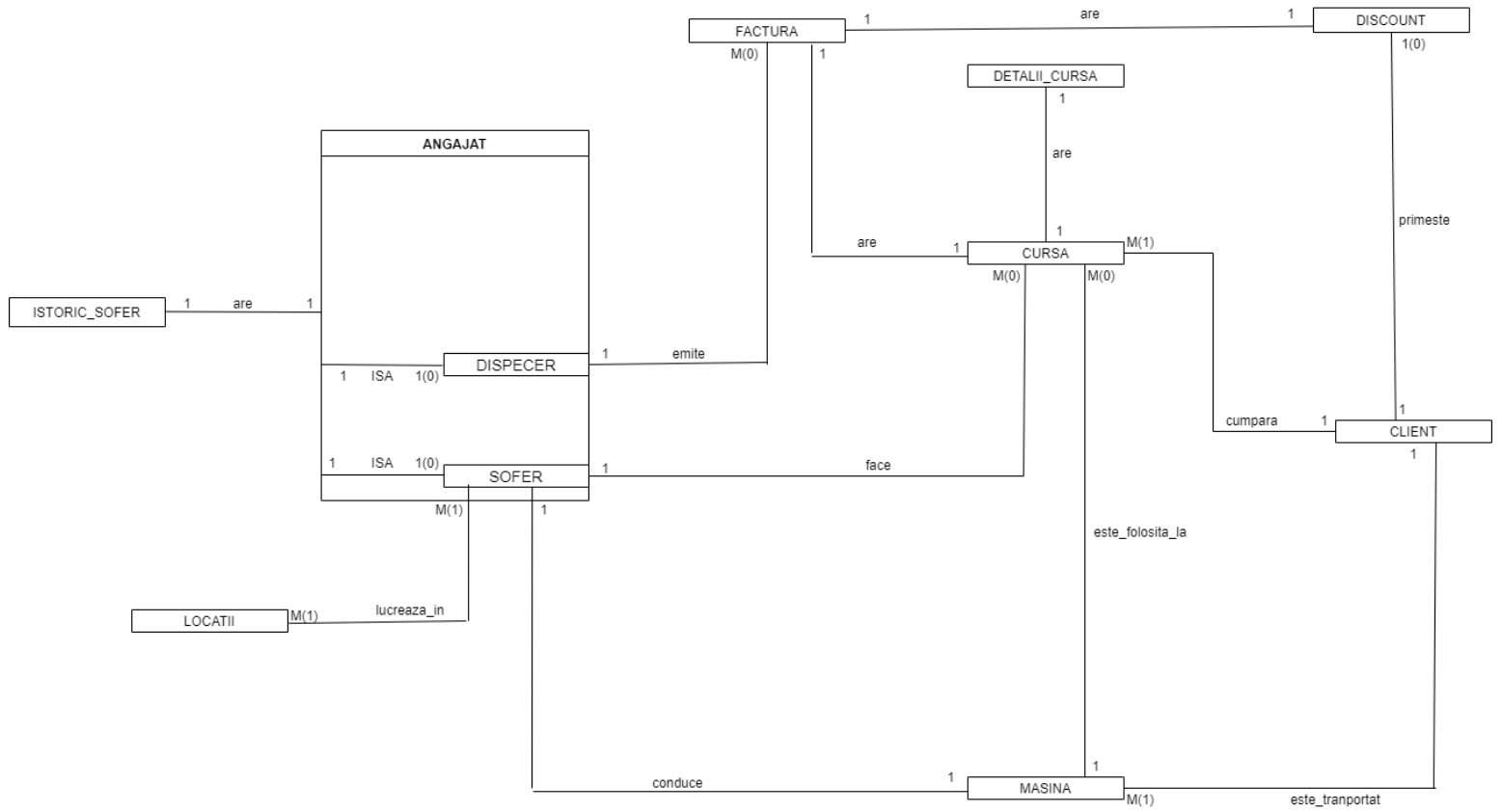
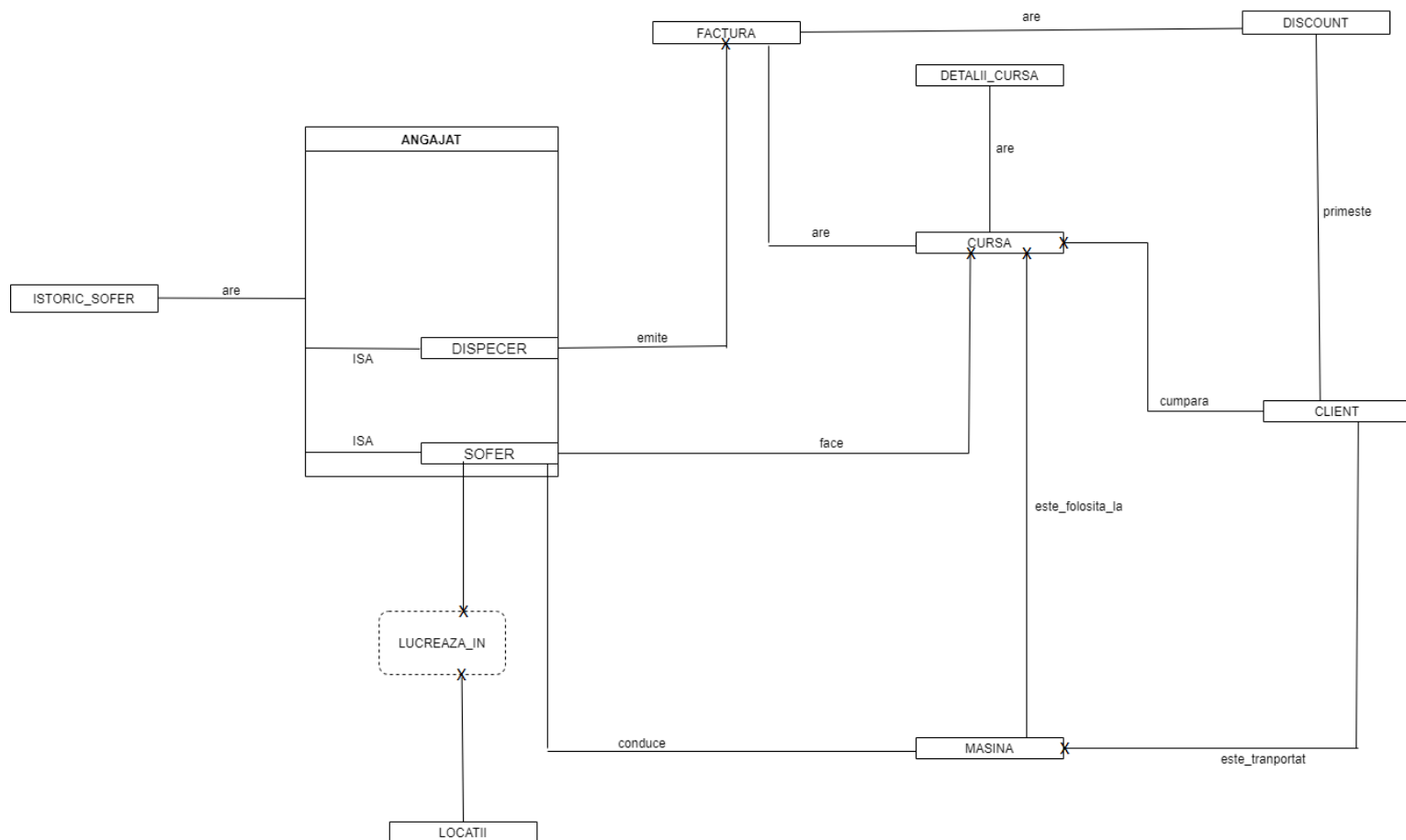


Diagrama conceptuală – Cerinta 3



Schemele relaționale 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)

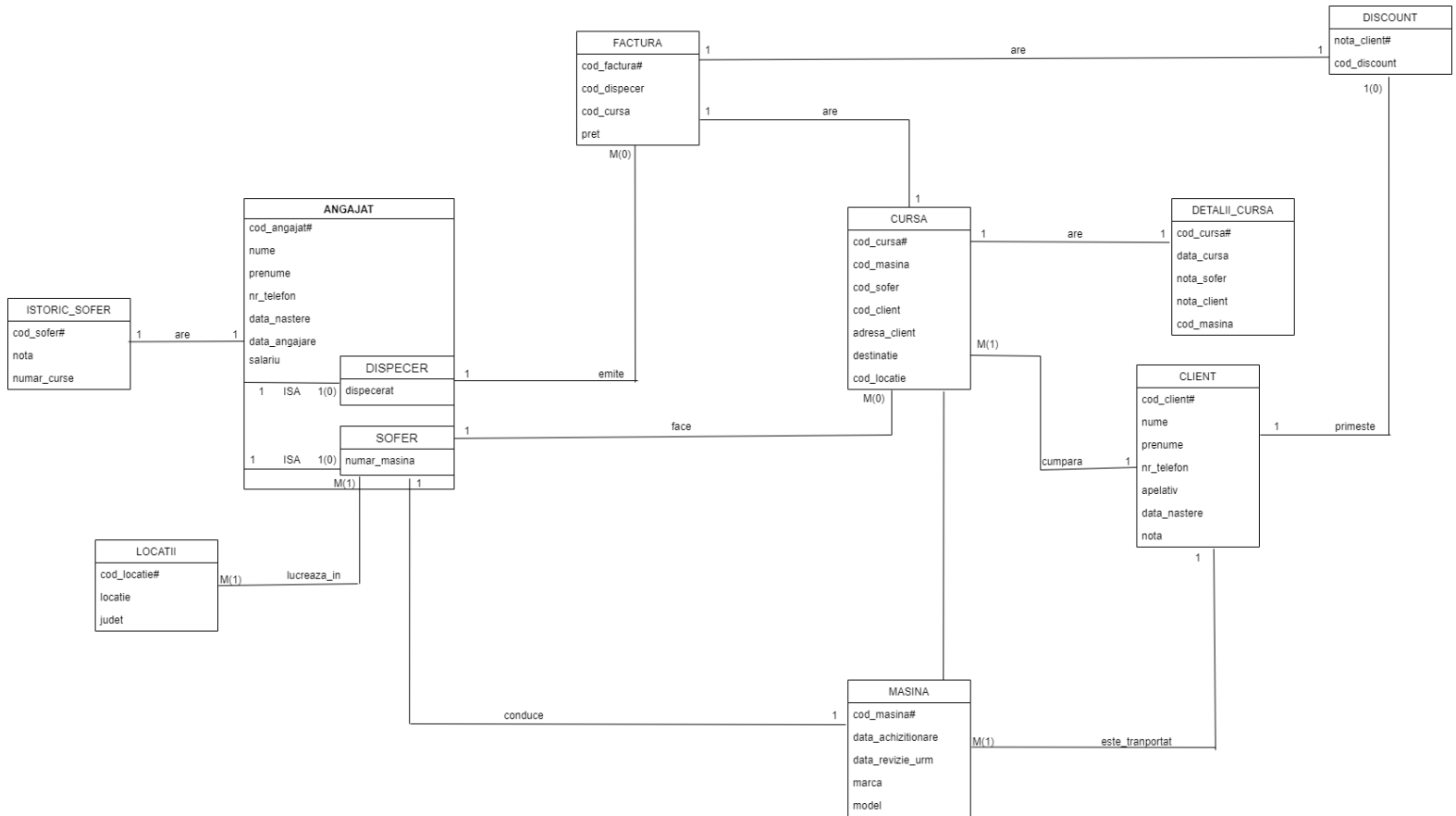
FACTURA(cod_factura#, cod_dispecer, cod_cursa, pret)

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 attribute



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_cursa 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,'Ilie','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);
```

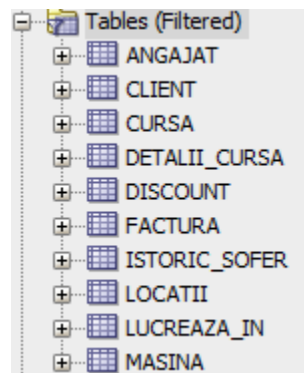
```
CREATE SEQUENCE ID_FACT
INCREMENT by 1
START WITH 1
MAXVALUE 1000
NOCYCLE;
INSERT INTO factura
VALUES(ID_FACT.NEXTVAL,105,2,34);
INSERT INTO factura
VALUES(ID_FACT.NEXTVAL,103,3,51.3);
INSERT INTO factura
VALUES(ID_FACT.NEXTVAL,104,4,21.1);
INSERT INTO factura
VALUES(ID_FACT.NEXTVAL,103,5,12.1);
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

◇ COD_ANGAJAT	◇ NUME	◇ PRENUME	◇ NR_TELEF...	◇ TIP_ANG...	◇ DATA_NASTERE	◇ DATA_ANGAJARE	◇ SALARIU	◇ NUMAR_MASINA	◇ DISPECERAT
101	Mar...	Cri...	074...	Sofer	21-MA...	08-SEP-17	3000	B 24 TAX	(null)
102	Petre	Ionel	072...	Sofer	27-AU...	01-OCT-18	3200	B 124 PEL	(null)
103	Pop	Alina	073...	Dis...	01-SE...	15-JAN-18	2500	(null)	Titan
104	Geo...	Damian	073...	Dis...	23-NO...	16-JUN-16	2500	(null)	Titan
105	Oprea	Teo...	024...	Dis...	12-DE...	08-FEB-20	2500	(null)	Dristor
106	Draoi	Rebeca	075...	Sofer	17-JU...	22-NOV-17	3200	IF 745 RBE	(null)
107	Marin	Fla...	074...	Sofer	07-JU...	07-SEP-19	3000	IS 24 FLV	(null)
108	Mocanu	Vlad	075...	Dis...	10-FE...	08-APR-21	2200	(null)	Dristor

Tabelul CLIENT

◇ COD_CLIENT	◇ NUME	◇ PRENUME	◇ NR_TELEFON	◇ APELATIV	◇ DATA_NASTERE	◇ NOTA
101	Martinescu	Vali	0724585435	Dl.	27-NOV-00	7
102	Popescu	Andrei	0213453567	(null)	12-APR-98	8
103	Mirea	Alexandra	0736459294	Dna.	13-MAY-88	10
104	Ilarie	David	0762469075	Dl.	01-MAY-88	9
105	Sima	Cezar	0216560666	Dl.	24-SEP-68	6
106	Petrea	Andreea	0213453567	Dra.	30-APR-92	9

Tabelul CURSA

◇ COD_CURSA	◇ COD_MASINA	◇ COD_SOFER	◇ COD_CLIENT	◇ ADRESA_CLIENT	◇ DESTINATIE	◇ COD_LOCATIE
2	B 24 TAX	101	101	Bulevardul Unirii 5	Calea Victoriei 118	100
3	B 124 PEL	102	101	Bulevardul Dacia 141	Bulevardul Libertatii 16	100
4	B 24 TAX	101	102	Aleea Privighetor...	Strada Polona 45	100
5	IF 745 RBE	106	105	Aleea Eprubetei 23	Strada Fizicienilor 55	100
6	B 24 TAX	101	103	Strada Toamnei 2	Strada Beirut 15	100
7	IF 745 RBE	106	104	Strada Caraiman 13	Calea Motilor 118	200
8	IF 745 RBE	106	104	Strada Spartacus 3	Strada Tudor Vladimir...	400
9	B 124 PEL	102	105	Strada Daliei 22	Strada Zorile 75	300

Tabelul DETALII_CURSA

❖ COD_CURSA	❖ DATA_CURSA	❖ NOTA_SOFER	❖ NOTA_CLIENT
2	07-APR-21	8	10
3	08-APR-21	6	7
4	10-APR-21	7	9
5	10-APR-21	9	9
6	10-APR-21	8	8
7	21-AUG-20	8	8
8	05-JUL-20	9	9
9	08-JAN-20	9	7

Tabelul DISCOUNT

❖ NOTA_DISCOUNT	❖ COD_DISCOUNT
10	10
9	7
8	5
7	2
6	1
5	0
4	0
3	0
2	0
1	0
0	0

Tabelul FACTURA

❖ COD_FACTURA	❖ COD_DISPECER	❖ COD_CURSA	❖ PRET
2	103	2	51.3
3	104	3	21.1
4	103	4	12.1
5	103	5	60
6	103	6	20
7	102	7	40
8	104	8	60

Tabelul ISTORIC_SOFER

COD_SOFER	NOTA	NUMAR_CURSE
101	7.66	3
102	8	1
106	7	1
107	10	0

Tabelul LOCATII

COD_LOCATIE	LOCALITATE	JUDET
100	Bucuresti	Bucuresti
200	Cluj-Napoca	Cluj
300	Timisoara	Timis
400	Oradea	Bihor
500	Iasi	Iasi
600	Alexandria	Teleorman

Tabelul LUCREAZA_IN

COD_ANG...	COD_LOCATIE
101	100
101	200
102	100
106	100
106	500
106	200
106	400
107	500
107	400
107	300

Tabelul MASINA

COD_MASINA	DATA_ACHIZITIONARE	DATA_REVIZIE_URM	MARCA	MODEL
B 24 TAX	01-JUN-08	20-MAR-22	Dacia	Logan
B 124 PEL	20-NOV-10	03-JAN-22	Skoda	Octavia
IF 745 RBE	13-DEC-14	15-DEC-21	Renault	Megane
B 123 TAX	01-OCT-20	07-MAY-22	Dacia	Logan
B 167 TAX	01-OCT-20	06-JUN-21	Dacia	Logan
IS 24 FLV	01-MAY-10	01-MAY-22	Chevrolet	Aveo

Normalizarea FN1-FN3

Tabelul ANGAJAT

Forma normala 1 (FN1)

O relație 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 relație R este în a doua formă normală dacă și numai dacă:

- relația 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:

- $\{\text{cod_angajat\#}\} \rightarrow \{\text{Nume}, \text{Tip_angajat}\}$
- $\{\text{cod_locatie\#}\} \rightarrow \{\text{Oras}\}$

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 relație R este în a treia formă normală dacă și numai dacă:

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

Astfel avem dependentele pentru relația LOCATIE_ANGAJAT:

- $\{\text{cod_angajat}\} \rightarrow \{\}$
- $\{\text{cod_locatie}\} \rightarrow \{\text{Oras}\}$

Se aplică regula Casey Delobel . Relația se descompune, prin eliminarea dependențelor funcționale tranzitive , în proiecțiile:

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:

```
1  --EX 6: Procedura care afiseaza toti angajatii si, daca anagajatul este sofer, afiseaza orasele in care poate conduce:
2  CREATE OR REPLACE PROCEDURE ex6
3  IS
4  TYPE tablou_imbricat IS TABLE OF NUMBER;
5  TYPE tablou_indexat IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
6  coduri_locatii tablou_imbricat := tablou_imbricat();
7  coduri_angajati tablou_indexat;
8  cod_ang NUMBER(4);
9  nr_ang NUMBER(3);
10 nume angajat.nume$type;
11 prenume angajat.prenume$type;
12 oras locatii.localitate$type;
13 rol_ang angajat.tip_angajat$type;
14 BEGIN
15 SELECT count(*) into nr_ang from angajat;
16 SELECT cod_angajat bulk collect into coduri_angajati from angajat;
17 for a in coduri_angajati.FIRST..coduri_angajati.LAST LOOP
18 SELECT nume,prenume,tip_angajat INTO nume,prenume,rol_ang FROM angajat WHERE cod_angajat=coduri_angajati(a);
19 cod_ang:=coduri_angajati(a);
20 SELECT cod_locatie bulk collect into coduri_locatii from lucreaza_in where cod_angajat=cod_ang;
21
22 DBMS_OUTPUT.PUT_LINE('Angajatul '||nume||' '||prenume||' cu codul '||cod_ang||':');
23 if rol_ang='Sofer' then
24 if coduri_locatii.count>0 then
25 DBMS_OUTPUT.PUT_LINE('Lucreaza in orasele:');
26 for i in coduri_locatii.FIRST..coduri_locatii.LAST LOOP
27 SELECT localitate INTO oras FROM locatii where cod_locatie=coduri_locatii(i);
28 DBMS_OUTPUT.PUT_LINE(oras);
29 END LOOP;
30 else DBMS_OUTPUT.PUT_LINE('Nu poate lucra in nici un oras');
31 end if;
32 else DBMS_OUTPUT.PUT_LINE('Nu este sofer');
33 end if;
34 DBMS_OUTPUT.NEW_LINE;
35 END LOOP;
36 END;
37 /
38 EXECUTE ex6;
```

Angajatul Marinescu Cristian cu codul 101:
Lucreaza in orasele:
Bucuresti
Cluj-Napoca

Angajatul Petre Ionel cu codul 102:
Lucreaza in orasele:
Bucuresti

Angajatul Pop Alina cu codul 103:
Nu este sofer

Angajatul Georgescu Damian cu codul 104:
Nu este sofer

Angajatul Oprea Teohari cu codul 105:
Nu este sofer

Angajatul Dragoi Rebeca cu codul 106:
Lucreaza in orasele:
Bucuresti
Cluj-Napoca
Oradea
Iasi

Angajatul Marin Flavius cu codul 107:
Lucreaza in orasele:
Timisoara
Oradea
Iasi

Angajatul Mocanu Vlad cu codul 108:
Nu este sofer

Angajatul Georgescu Matei cu codul 109:
Nu este sofer

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:

```

40 |
41 | --EX 7: Procedura care, la executie, afiseaza toate masinile, soferul lor si nr de curse facute (daca au):
42 | CREATE OR REPLACE PROCEDURE ex7
43 | IS
44 |   nr_curse NUMBER(4);
45 |   cursor masina_cursor is
46 |   select cod_masina, marca, model
47 |   from masina;
48 |   CURSOR angajat_cursor ( cod_masina masina.cod_masina%type) IS
49 |   select nume, prenume, salariu
50 |   from angajat
51 |   where numar_masina = cod_masina;
52 |
53 | BEGIN
54 |   for m in masina_cursor loop
55 |     DBMS_OUTPUT.PUT_LINE('-----');
56 |     DBMS_OUTPUT.PUT_LINE( 'Masina ' || m.cod_masina || ' ' || m.marca || ' ' || m.model);
57 |     DBMS_OUTPUT.PUT_LINE('-----');
58 |     SELECT count(*) into nr_curse from cursa where cod_masina=m.cod_masina;
59 |     if nr_curse =0 then
60 |       DBMS_OUTPUT.PUT_LINE( 'Aceasta masina nu a facut curse');
61 |     else
62 |       for a in angajat_cursor(m.cod_masina) loop
63 |         DBMS_OUTPUT.PUT_LINE('Sofer: ' || a.nume || ' ' || a.prenume || ', Numar curse: ' || nr_curse);
64 |         DBMS_OUTPUT.PUT_LINE('-----');
65 |       end loop;
66 |     end if;
67 |     DBMS_OUTPUT.NEW_LINE;
68 |   end loop;
69 | end;
70 | /
71 |
72 |
73 | EXECUTE ex7;
74 |

```

```

Masina B 24 TAX Dacia Logan
-----
Sofer: Marinescu Cristian, Numar curse: 4
-----

Masina B 124 PEL Skoda Octavia
-----
Sofer: Petre Ionel, Numar curse: 2
-----

Masina IF 745 RBE Renault Megane
-----
Sofer: Dragoi Rebeca, Numar curse: 3
-----

Masina B 123 TAX Dacia Logan
-----
Aceasta masina nu a facut curse
-----

Masina B 167 TAX Dacia Logan
-----
Aceasta masina nu a facut curse
-----

Masina IS 24 FLV Chevrolet Aveo
-----
Aceasta masina nu a facut curse
-----

```

Procedure EX7 compiled

PL/SQL procedure successfully completed.

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:

```
Worksheet | Query Builder
73
74 --EX 8: Functie care primeste ca parametru numele angajatului si afiseaza detalii despre acesta si in functie de tipul
75 --de angajat, numarul masinii, numarul de curse, nr de telefon si masina sau dispeceratul si numarul de facturi emise:
76 CREATE OR REPLACE FUNCTION ex8 (nume_ang IN angajat.nume%TYPE) RETURN VARCHAR2 IS
77 rezultat VARCHAR2(200); cod_ang angajat.cod_angajat%TYPE; prenume angajat.prenume%TYPE; tip_ang angajat.tip_angajat%TYPE;
78 sal angajat.salariu%TYPE; nr_masina angajat.numar_masina%TYPE; dispecerat angajat.dispecerat%TYPE;
79 nr NUMBER(4); oras locatii.localitate%TYPE; nr_telefon angajat.nr_telefon%TYPE;
80 nota istoric_sofer.nota%TYPE; marca masina.marca%TYPE; model masina.model%TYPE;
81 BEGIN
82 BEGIN
83 select cod_angajat, prenume, tip_angajat, salariu, numar_masina, dispecerat
84 into cod_ang, prenume, tip_ang, sal, nr_masina, dispecerat
85 from angajat
86 where upper(nume_ang) = upper(nume);
87 EXCEPTION WHEN NO_DATA_FOUND
88 THEN DBMS_OUTPUT.PUT_LINE('Angajatul nu exista'); RAISE_APPLICATION_ERROR (-20000, 'Angajatul nu exista'); END;
89 rezultat:=rezultat||nume_ang||' '||prenume||' '||tip_ang||' , salariu '||CAST(sal as VARCHAR2);
90 if tip_ang='Sofer' then
91 SELECT count(*) INTO nr FROM cursa where cod_sofer=cod_ang group by cod_sofer;
92 SELECT a.nr_telefon, i.nota, m.marca, m.model into nr_telefon, nota, marca, model
93 from angajat a join istoric_sofer i on a.cod_angajat=i.cod_sofer join masina m on a.numar_masina=m.cod_masina
94 where cod_ang=a.cod_angajat;
95 rezultat:=rezultat||' , numar masina '||nr_masina||' , nr curse '||CAST(nr as VARCHAR2)||' , nr telefon '||nr_telefon
96 ||' , nota sofer '||nota||' , masina '||marca||' '||model;
97 elsif tip_ang='Dispecer' then
98 SELECT count(*) INTO nr FROM factura where cod_dispecer=cod_ang group by cod_dispecer;
99 rezultat:=rezultat||' , dispecerat '||dispecerat||' , nr facturi '||CAST(nr as VARCHAR2);
100 else RETURN 'Angajatul nu este sofer sau dispecer';
101 end if;
102 RETURN rezultat;
103 EXCEPTION
104 WHEN NO_DATA_FOUND THEN
105 DBMS_OUTPUT.PUT_LINE('Angajatul nu are curse realizate sau facturi emise');
106 RAISE_APPLICATION_ERROR (-20001, 'Angajatul nu are curse realizate sau facturi emise');
107 WHEN TOO_MANY_ROWS THEN
108 DBMS_OUTPUT.PUT_LINE('Mai multi angajati cu acelasi nume');
109 RAISE_APPLICATION_ERROR (-20002, 'Mai multi angajati cu acelasi nume');
110 END;
111
Script Output | Query Result
Task completed in 0.122 seconds

Function EX8 compiled
```

```

112 BEGIN
113 DBMS_OUTPUT.PUT_LINE(ex8('Pop'));
114 END;
115 /
116 BEGIN
117 DBMS_OUTPUT.PUT_LINE(ex8('Marinescu'));
118 END;
119 /
120 BEGIN
121 DBMS_OUTPUT.PUT_LINE(ex8('Georgescu'));
122 END;
123 /
124 BEGIN
125 DBMS_OUTPUT.PUT_LINE(ex8('Marin'));
126 END;
127 /
128 BEGIN
129 DBMS_OUTPUT.PUT_LINE(ex8('Stefanescu'));
130 END;

```

Pop Alina, Dispecer, salariu 2500, dispecerat Titan, nr facturi 4

Marinescu Cristian, Sofer, salariu 3000, numar masina B 24 TAX, nr curse 4, nr telefon 0742536126, nota sofer 7.67, masina Dacia Loga

Mai multi angajati cu acelasi nume

Angajatul nu are curse realizate sau facturi emise

Angajatul nu exista

ORA-20002: Mai multi angajati cu acelasi nume
ORA-06512: at "ALEX.EX8", line 34
ORA-06512: at line 2

Error starting at line : 124 in command -
BEGIN
DBMS_OUTPUT.PUT_LINE(ex8('Marin'));
END;
Error report -
ORA-20001: Angajatul nu are curse realizate sau factur
ORA-06512: at "ALEX.EX8", line 31
ORA-06512: at line 2

Error starting at line : 128 in command -
BEGIN
DBMS_OUTPUT.PUT_LINE(ex8('Stefanescu'));
END;
Error report -
ORA-20000: Angajatul nu exista

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 locatie.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:

```
133 --EX 9: O procedura care primeste ca parametru un nume de strada si afiseaza detalii din mai multe tabele despre cursa,
134 --masina sau sofer:
135 CREATE OR REPLACE PROCEDURE ex9
136 (adresa_c IN cursa.adresa_client%type)
137 AS
138 id_cursa cursa.cod_cursa%type; cod_sofer cursa.cod_sofer%type; v_adresa cursa.adresa_client%type;
139 destinatie cursa.adresa_client%type; cod_loc cursa.cod_locatie%type; pret factura.pret%type;
140 v_masina masina.cod_masina%type; marca masina.marca%type; v_model masina.model%type;
141 nota_sofer detalii_cursa.nota_sofer%type; localit locatii.localitate%type; nume angajat.nume%type; prenume angajat.prenume%type;
142 BEGIN
143 SELECT cod_cursa into id_cursa
144 from cursa where UPPER(adresa_client) LIKE '%'||UPPER(adresa_c)||'%';
145 SELECT c.cod_sofer,c.adresa_client,c.destinatie,c.cod_locatie,f.pret,a.numar_masina,a.nume,a.prenume,
146 m.marca,m.model, d.nota_sofer, l.localitate into cod_sofer,v_adresa,destinatie,cod_loc,pret,v_masina,nume,prenume,
147 marca,v_model,nota_sofer,localit from cursa c join factura f on c.cod_cursa=f.cod_cursa join angajat a on
148 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
149 join locatii l on l.cod_locatie=c.cod_locatie
150 where c.cod_cursa=id_cursa;
151 DBMS_OUTPUT.PUT_LINE('Cursa ' || id_cursa || ' din orasul ' || localit || ' de la adresa ' || v_adresa || ' la adresa ' ||
152 destinatie || chr(10)|| ' a fost facuta cu masina ' || marca || ' ' || v_model || ', soferul a primit nota ' || nota_sofer
153 || ' iar pretul cursei a fost ' || pret);
154 EXCEPTION
155 WHEN NO_DATA_FOUND THEN
156 DBMS_OUTPUT.PUT_LINE('Cursa care porneste de la adresa specificata nu exista');
157 RAISE_APPLICATION_ERROR (-20000,'Cursa care porneste de la adresa specificata nu exista');
158 WHEN TOO_MANY_ROWS THEN
159 DBMS_OUTPUT.PUT_LINE('Mai multe curse cu aceeasi locatie, dati o locatie mai specifica');
160 RAISE_APPLICATION_ERROR (-20001, 'Mai multe curse cu aceeasi locatie, dati o locatie mai specifica');
161 END;
162 /
163 EXECUTE ex9('Unirii');
164 EXECUTE ex9('Dacia');
165 EXECUTE ex9('Rahovei');
```

Procedure EX9 compiled

```
163 EXECUTE ex9('Unirii');
164 EXECUTE ex9('Dacia');
165 EXECUTE ex9('Rahovei');
```

Cursa 2 din orasul Bucuresti de la adresa Bulevardul Unirii 5 la adresa Calea Victoriei 118 a fost facuta cu masina Dacia Logan, soferul a primit nota 8 iar pretul cursei a fost 34

Mai multe curse cu aceeasi locatie, dati o locatie mai specifica

Cursa care porneste de la adresa specificata nu exista

Procedure EX9 compiled

PL/SQL procedure successfully completed.

Error starting at line : 164 in command -
BEGIN ex9('Dacia'); END;
Error report -
ORA-20001: Mai multe curse cu aceeasi locatie, dati o locatie mai specifica
ORA-06512: at "ALEX.EX9", line 26
ORA-06512: at line 1

Error starting at line : 165 in command -
BEGIN ex9('Rahovei'); END;
Error report -
ORA-20000: Cursa care porneste de la adresa specificata nu exista
ORA-06512: at "ALEX.EX9", line 23

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.

```
32 UPDATE detalii_cursa set nota_sofer=8,nota_client=8 where cod_cursa=7;
33
```



Script Output x Query Result x

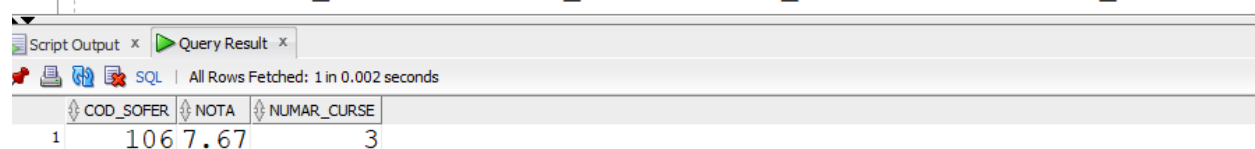
All Rows Fetched: 1 in 0.001 seconds

	COD_SOFER	NOTA	NUMAR_CURSE
1	106	8.67	3

	COD_CLIENT	NUME	PRENUME	NR_TELEFON	APELATIV	DATA_NASTERE	NOTA
1	104	Ilarie	David	0762469075	Dl.	01-MAY-88	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.

```
32 UPDATE detalii_cursa set nota_sofer=5,nota_client=5 where cod_cursa=7;
```



Script Output x Query Result x

All Rows Fetched: 1 in 0.002 seconds

	COD_SOFER	NOTA	NUMAR_CURSE
1	106	7.67	3

	COD_CLIENT	NUME	PRENUME	NR_TELEFON	APELATIV	DATA_NASTERE	NOTA
1	104	Ilarie	David	0762469075	Dl.	01-MAY-88	7

Cerinta 11

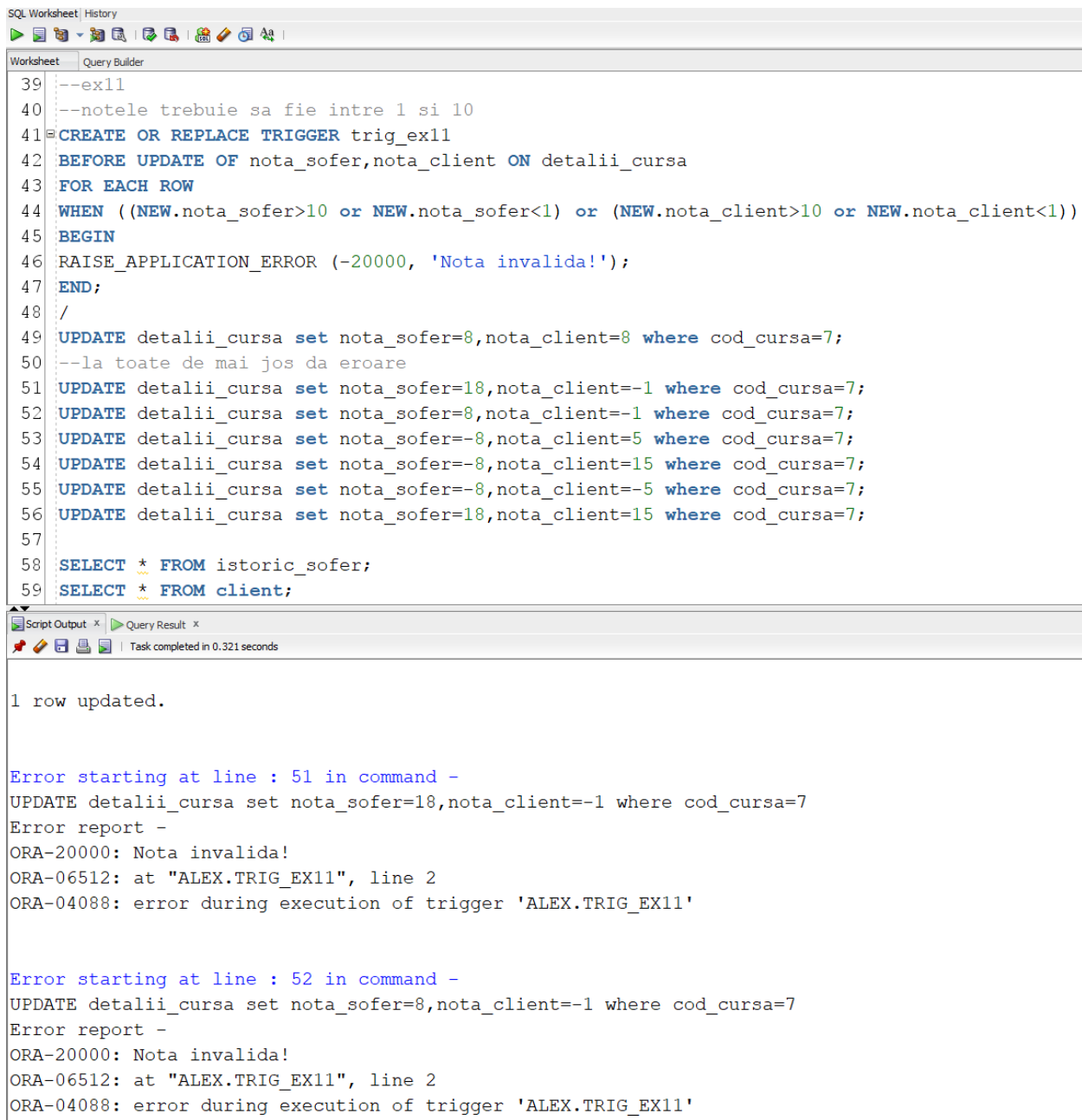
Creati un trigger care impune ca notele trebuie sa fie intre 1 si 10:

```
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:



The screenshot displays an SQL Worksheet application. The top section, titled 'Worksheet' and 'Query Builder', contains a SQL script. The script starts with a comment '--ex11' and another '--notele trebuie sa fie intre 1 si 10'. It then creates a trigger named 'trig_ex11' that fires before updates on the 'detalii_cursa' table. The trigger logic checks if the new 'nota_sofer' or 'nota_client' values are outside the range of 1 to 10. If so, it raises an application error with the message 'Nota invalida!'. Following the trigger creation, there are several UPDATE statements for 'detalii_cursa' with various values for 'nota_sofer' and 'nota_client' where 'cod_cursa=7'. The script concludes with two SELECT statements: one from 'istoric_sofer' and another from 'client'.

The bottom section of the interface shows the execution results. It indicates that 1 row was updated. Below this, there are two error messages. The first error occurs at line 51 of the command, which is the UPDATE statement setting 'nota_sofer=18' and 'nota_client=-1'. The error report shows 'ORA-20000: Nota invalida!', 'ORA-06512: at "ALEX.TRIG_EX11", line 2', and 'ORA-04088: error during execution of trigger "ALEX.TRIG_EX11"'. The second error is identical and occurs at line 52, which is the UPDATE statement setting 'nota_sofer=8' and 'nota_client=-1'.

```
39 --ex11
40 --notele trebuie sa fie intre 1 si 10
41 CREATE OR REPLACE TRIGGER trig_ex11
42 BEFORE UPDATE OF nota_sofer,nota_client ON detalii_cursa
43 FOR EACH ROW
44 WHEN ((NEW.nota_sofer>10 or NEW.nota_sofer<1) or (NEW.nota_client>10 or NEW.nota_client<1))
45 BEGIN
46 RAISE_APPLICATION_ERROR (-20000, 'Nota invalida!');
47 END;
48 /
49 UPDATE detalii_cursa set nota_sofer=8,nota_client=8 where cod_cursa=7;
50 --la toate de mai jos da eroare
51 UPDATE detalii_cursa set nota_sofer=18,nota_client=-1 where cod_cursa=7;
52 UPDATE detalii_cursa set nota_sofer=8,nota_client=-1 where cod_cursa=7;
53 UPDATE detalii_cursa set nota_sofer=-8,nota_client=5 where cod_cursa=7;
54 UPDATE detalii_cursa set nota_sofer=-8,nota_client=15 where cod_cursa=7;
55 UPDATE detalii_cursa set nota_sofer=-8,nota_client=-5 where cod_cursa=7;
56 UPDATE detalii_cursa set nota_sofer=18,nota_client=15 where cod_cursa=7;
57
58 SELECT * FROM istoric_sofer;
59 SELECT * FROM client;
```

1 row updated.

Error starting at line : 51 in command -
UPDATE detalii_cursa set nota_sofer=18,nota_client=-1 where cod_cursa=7
Error report -
ORA-20000: Nota invalida!
ORA-06512: at "ALEX.TRIG_EX11", line 2
ORA-04088: error during execution of trigger 'ALEX.TRIG_EX11'

Error starting at line : 52 in command -
UPDATE detalii_cursa set nota_sofer=8,nota_client=-1 where cod_cursa=7
Error report -
ORA-20000: Nota invalida!
ORA-06512: at "ALEX.TRIG_EX11", line 2
ORA-04088: error during execution of trigger 'ALEX.TRIG_EX11'

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:
```

```
61 --ex 12
62 --trigger care interzice drop la tabele
63 CREATE OR REPLACE TRIGGER trig_ex12
64 AFTER DROP ON SCHEMA
65 BEGIN
66     RAISE_APPLICATION_ERROR (-20000, 'Nu se poate da drop la tabelele existente!');
67     ROLLBACK;
68 END;
69 /
70 DROP TRIGGER trig_ex12;
71
72 CREATE TABLE test_trig_ex12(
73 id NUMBER(2)
74 );
75
76 DROP TABLE test_trig_ex12;
77
```

Script Output x Query Result x

Task completed in 0.326 seconds

Error starting at line : 76 in command -
DROP TABLE test_trig_ex12

Error report -
ORA-00604: error occurred at recursive SQL level 1
ORA-20000: Nu se poate da drop la tabelele existente!
ORA-06512: at line 2
00604. 00000 - "error occurred at recursive SQL level %s"
*Cause: An error occurred while processing a recursive SQL statement
(a statement applying to internal dictionary tables).
*Action: If the situation described in the next error on the stack
can be corrected, do so; otherwise contact Oracle Support.

Cerinta 13

CREATE OR REPLACE PACKAGE pachet1 is

PROCEDURE ex6;

PROCEDURE ex7;

FUNCTION ex8(ume_ang IN angajat.ume%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(ume_ang) = upper(ume);
EXCEPTION
WHEN NO_DATA_FOUND
    THEN
        DBMS_OUTPUT.PUT_LINE('Angajatul nu exista');
        RAISE_APPLICATION_ERROR (-20000,'Angajatul nu exista');
END;

rezultat:=rezultat||ume_ang||' '||preume||', '||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;

elseif 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 locatie.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 locatie l on l.cod_locatie=c.cod_locatie

where c.cod_cursa=id_cursa;

```

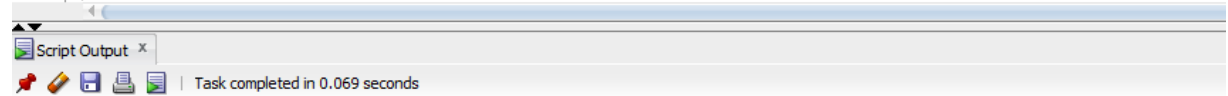
DBMS_OUTPUT.PUT_LINE('Cursa ' || id_cursa || ' din orasul ' || localit || ' de la adresa ' ||
v_adresa || ' la adresa ' ||
destinatie || chr(10) || ' a fost facuta cu masina ' || marca || ' ' || v_model || ', soferul a primit nota ' ||
nota_sofer
|| ' iar pretul cursei a fost ' || pret);
EXCEPTION
WHEN 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:

```
1 CREATE OR REPLACE PACKAGE pachet1 is
2
3 PROCEDURE ex6;
4
5 PROCEDURE ex7;
6
7 FUNCTION ex8(nume_ang IN angajat.nume%type)
8 RETURN VARCHAR2;
9
10 PROCEDURE ex9(adresa_c IN cursa.adresa_client%type);
11
12 END pachet1;
13 /
14
15 CREATE OR REPLACE PACKAGE BODY pachet1 is
16
17 --exercitiul 6
18 --Creati o procedura care, la executie, afiseaza toti angajatii si,
19 PROCEDURE ex6
20
21 IS
22 TYPE tablou_imbricat IS TABLE OF NUMBER;
23 TYPE tablou_indexat IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
24
25 coduri_locatii tablou_imbricat := tablou_imbricat();
26 coduri_angajati tablou_indexat;
27
28 cod_ang NUMBER(4);
29 nr_ang NUMBER(3);
30 nume angajat.nume%type;
31 prenume angajat.prenume%type;
32 oras locatii.localitate%type;
33 rol_ang angajat.tip_angajat%type;
34 BEGIN
```



Package PACHET1 compiled

Package Body PACHET1 compiled