

Studiu de caz

Exemplele din acest capitol se referă la proiectarea unui model de date ce furnizează informații despre service-urile auto din România, un loc ce este constant frecventat, deoarece numărul mașinilor din România este în continuă creștere, iar acestea trebuie întreținute și reparate constant.

Voi prezenta modelul de date, restricțiile pe care trebuie să le respecte și voi construi diagrama E/R corespunzătoare, dar și diagrama conceptuală.

Modelul de date va gestiona informații legate de organizarea și funcționarea service-urilor auto din România. Toate acestea au furnizori - ei se află în colaborare cu service-uri pentru a le aproviziona cu piese auto atunci când este nevoie (în baza unui contract). Un anumit tip de piesă auto poate fi furnizată de un singur furnizor.

Fiecare service se află într-o anumită locație și are unul sau mai mulți angajați. Un service nu se poate afla decât într-o locație și numai una.

Angajații au fiecare câte un job (mecanic, electrician auto, tinichigiu) și fiecare este arondat sa monteze piese. Clienții își fac o programare și vin cu o anumită mașină.

Fiecare mașină care vine în service are nevoie de minim o piesă, care este montată de unul sau de mai mulți angajați.

Modelul de date respectă anumite restricții de funcționare:

- Un angajat poate monta mai multe piese și minim una, iar o piesă trebuie să fie montată de minim un angajat.
- Furnizorii aprovizionează service-ul auto cu piese auto, vopsea și diferite materiale pe durata contractului.
- Fiecare angajat trebuie să aibă un job și numai unul.
- O piesă este furnizată de un singur furnizor, care poate furniza mai mult de o piesă.

Entități

Pentru modelul de date referitor la service-urile auto din România, structurile LOCATIE, SERVICES, ANGAJAT, JOBS, PIESA, MASINA, CLIENTS, PROGRAMARE, FURNIZOR, CONTRACT reprezintă entități.

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

Toate entitățile sunt independente.

SERVICES = loc, firmă care se ocupă cu repararea, vopsirea și îngrijirea mașinilor, vehiculelor auto. Cheia primară a entității este *id_service*.

LOCATIE = entitate care identifică locația unui service auto. Cheia primară este *id_locatie*.

ANGAJAT = persoană fizică, angajată de către un service auto în baza unor cunoștințe pentru a repara, vopsi sau îngriji autovehiculele clienților care vin la service. Cheia primară a entității este *id_angajat*.

JOBS = identifică specializarea pe care o are un angajat. Cheia primară a entității este *id_job*.

PIESA = entitate ce reprezintă piesa care este defectă la mașina venită în service și este înlocuită pe aceasta de către un angajat. Cheia primară a entității este *id_piesa*.

MASINA = entitate ce semnifică autovehiculul adus de către un client pentru a fi reparat la service. Cheia primară a entității este *id_masina*.

CLIENTS = persoană fizică sau juridică care dorește repararea mașinii sau a mașinilor la service. Cheia primară a entității este *id_client*.

PROGRAMARE = reprezintă datele unei programări la service-ul auto realizate de către un client. Cheia primară a entității este *id_programare*.

FURNIZOR = persoană juridică, firmă specializată, care încheie un contract cu service-urile auto pentru a le aproviziona cu piese auto, vopsea auto și alte materiale necesare. Cheia primară a entității este *id_furnizor*.

CONTRACT = entitate care semnifică tipul și valabilitatea unui contract încheiat de un service cu un furnizor. Cheia primară a entității este *id_contract*.

Relații

Voi 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.

SERVICES_se_afla_LOCATIE = relație care leagă entitățile *SERVICES* și *LOCATIE*, reflectând legătura dintre acestea (ce locație are un anumit service auto). Ea are cardinalitatea minimă și cea maximă 1:1 (un service se află în exact o locație).

SERVICES_are_ANGAJAT = relație care leagă entitățile *SERVICES* și *ANGAJAT*, reflectând legătura dintre acestea (angajații lucrează la un singur service). Relația are cardinalitatea minimă 1:1 (un angajat trebuie să lucreze în cel puțin un service și un service trebuie să aibă cel puțin un angajat) și cardinalitatea maximă 1:n (un service poate avea mai mulți angajați, dar un angajat poate lucra într-un singur service).

ANGAJAT_are_JOBS = relație care leagă entitățile ANGAJAT și JOBS, reflectând legătura dintre acestea (fiecare angajat trebuie să aibă un job și exact unul, iar un job poate fi practicat de mai mulți angajați). Relația are cardinalitatea minimă 1:1 (orice angajat are nevoie de minim un job și orice job trebuie să fie practicat de măcar un angajat) și cardinalitatea maximă 1:n (un job poate fi practicat de mai mulți angajați, dar un angajat poate practica cel mult un job).

ANGAJAT_monteaza_PIESA = relație care leagă entitățile ANGAJAT și PIESA, reflectând legătura dintre acestea (angajații montează piese). Relația are cardinalitatea minimă 1:1 (un angajat trebuie să monteze cel puțin o piesă și o piesă trebuie montată de cel puțin un angajat) și cardinalitatea maximă m:n (o piesă poate fi montată de mai mulți angajați și un angajat poate monta mai multe piese).

FURNIZOR_furnizeaza_PIESA = relație care leagă entitățile FURNIZOR și PIESA, reflectând legătura dintre acestea (furnizorul aprovizionează service-ul cu piese auto). Relația are cardinalitatea minimă 1:1 și cardinalitatea maximă n:1.

FURNIZOR_semneaza_CONTRACT = relație care leagă entitățile FURNIZOR și CONTRACT, reflectând legătura dintre acestea (pentru a aproviziona service-urile, furnizorii trebuie să semneze un contract). Relația are cardinalitatea minimă și maximă 1:1.

CLIENTS_face_PROGRAMARE = relație care leagă entitățile CLIENTS și PROGRAMARE, reflectând legătura dintre acestea (pentru a se programa la service, un client trebuie să facă o programare). Relația are cardinalitatea minimă 1:1 și cardinalitatea maximă 1:n.

ANGAJAT_repara_MASINA_pentru_CLIENTS = relație de tip 3 ce leagă entitățile ANGAJAT, MASINA și CLIENTS, reflectând cine este angajatul care repară o anumită mașină pentru un anumit client (clientul poate veni cu mai multe mașini sau cu mașini diferite în service). Denumirea acestei relații va fi *repara*.

Atribute

Entitatea independentă SERVICES are ca atribute:

id_service = variabilă de tip întreg, de lungime maximă 6, care reprezintă codul unui service.

nume_service = variabilă de tip caracter, de lungime maximă 40, care reprezintă numele service-ului auto.

an_infiintare = variabilă de tip întreg, de lungime maximă 4, care reprezintă anul înființării service-ului.

id_locatie = variabilă de tip întreg, de lungime maximă 6, care reprezintă codul locației service-ului. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul LOCATIE.

Entitatea independentă LOCATIE are ca attribute:

id_locatie = variabilă de tip întreg, de lungime maximă 6, care reprezintă codul unei locații.

judet = variabilă de tip caracter, de lungime maximă 20, care reprezintă numele județului.

localitate = variabilă de tip caracter, de lungime maximă 30, care reprezintă numele localității.

strada = variabilă de tip caracter, de lungime maximă 30, care reprezintă numele străzii.

numar = variabilă de tip întreg, de lungime maximă 3, care reprezintă numărul străzii.

cod_postal = variabilă de tip caracter, de lungime maximă 6, care reprezintă codul postal în raza căruia se află service-ul.

Entitatea independentă ANGAJAT are ca attribute:

id_angajat = variabilă de tip întreg, de lungime maximă 6, care reprezintă codul unui angajat.

nume_angajat = variabilă de tip caracter, de lungime maximă 30, care reprezintă numele angajatului.

prenume_angajat = variabilă de tip caracter, de lungime maximă 30, care reprezintă prenumele angajatului.

salariu = variabilă de tip întreg, de lungime maximă 6, care reprezintă salariul unui angajat.

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

id_service = variabilă de tip întreg, de lungime maximă 6, care reprezintă codul service-ului la care lucrează angajatul. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul SERVICES.

id_job = variabilă de tip întreg, de lungime maximă 6, care reprezintă codul job-ului angajatului. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul JOBS.

Entitatea independentă JOBS are ca attribute:

id_job = variabilă de tip întreg, de lungime maximă 6, care reprezintă codul unui job.

nume_job = variabilă de tip caracter, de lungime maximă 30, care reprezintă numele jobului.

salariu_min = variabilă de tip întreg, de lungime maximă 6, care reprezintă salariul minim al unui job.

salariu_max = variabilă de tip întreg, de lungime maximă 6, care reprezintă salariul maxim al unui job.

Entitatea independentă PIESA are ca atribute:

id_piesa = variabilă de tip întreg, de lungime maximă 6, care reprezintă codul unei piese.

nume_piesa = variabilă de tip caracter, de lungime maximă 35, care reprezintă numele piesei.

fabricata_de = variabila de tip caracter, de lungime maxima 40, care reprezinta numele firmei care a fabricat piesa.

id_furnizor = variabila de tip întreg, de lungime maximă 6, care reprezintă codul furnizorului care a aprovizionat service-ul cu aceasta piesă. Atributul trebuie sa corespundă la o valoare a cheii primare din tabelul FURNIZOR.

Entitatea independentă CLIENTS are ca atribute:

id_client = variabilă de tip întreg, de lungime maximă 6, care reprezintă codul unui client.

nume_client = variabilă de tip caracter, de lungime maximă 30, care reprezintă numele clientului.

prenume_client = variabilă de tip caracter, de lungime maximă 30, care reprezintă prenumele clientului.

numar_telefon = variabilă de tip caracter, de lungime maximă 13, care reprezintă numărul de telefon al clientului.

Entitatea independentă PROGRAMARE are ca atribute:

id_programare = variabilă de tip întreg, de lungime maximă 6, care reprezintă codul unui programare.

data_programare = variabilă de tip dată calendaristică, care reprezintă data programării.

id_client = variabilă de tip întreg, de lungime maximă 6, care reprezintă codul clientului care a făcut programarea. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul CLIENTS.

Entitatea independentă MASINA are ca atribute:

id_masina = variabilă de tip întreg, de lungime maximă 6, care reprezintă codul unei mașini.

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

an_fabricatie = variabilă de tip întreg, de lungime maximă 4, care reprezintă anul în care a fost fabricată mașina.

Entitatea independentă FURNIZOR are ca attribute:

id_furnizor = variabilă de tip întreg, de lungime maximă 6, care reprezintă codul unui furnizor.

nume_furnizor = variabilă de tip caracter, de lungime maximă 40, care reprezintă numele furnizorului.

valoare = variabilă de tip întreg, de lungime maximă 8, care reprezintă suma limită pe care service-ul este dispus să o ofere pe durata contractului.

id_contract = variabilă de tip întreg, de lungime maximă 6, care reprezintă codul contractului înțocmit cu service-ul. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul CONTRACT.

Entitatea independentă CONTRACT are ca attribute:

id_contract = variabilă de tip întreg, de lungime maximă 6, care reprezintă codul unui contract.

data_incepere = variabilă de tip dată calendaristică, care reprezintă data începerii contractului.

durata = variabilă de tip numeric(real), de lungime maximă 6, dintre care 2 zecimale, care reprezintă durata contractului. Valoare implicită NULL – perioadă nedeterminată.

GESTIUNEA SERVICE-URILOR AUTO DIN ROMÂNIA

Diagrama entitate – relație

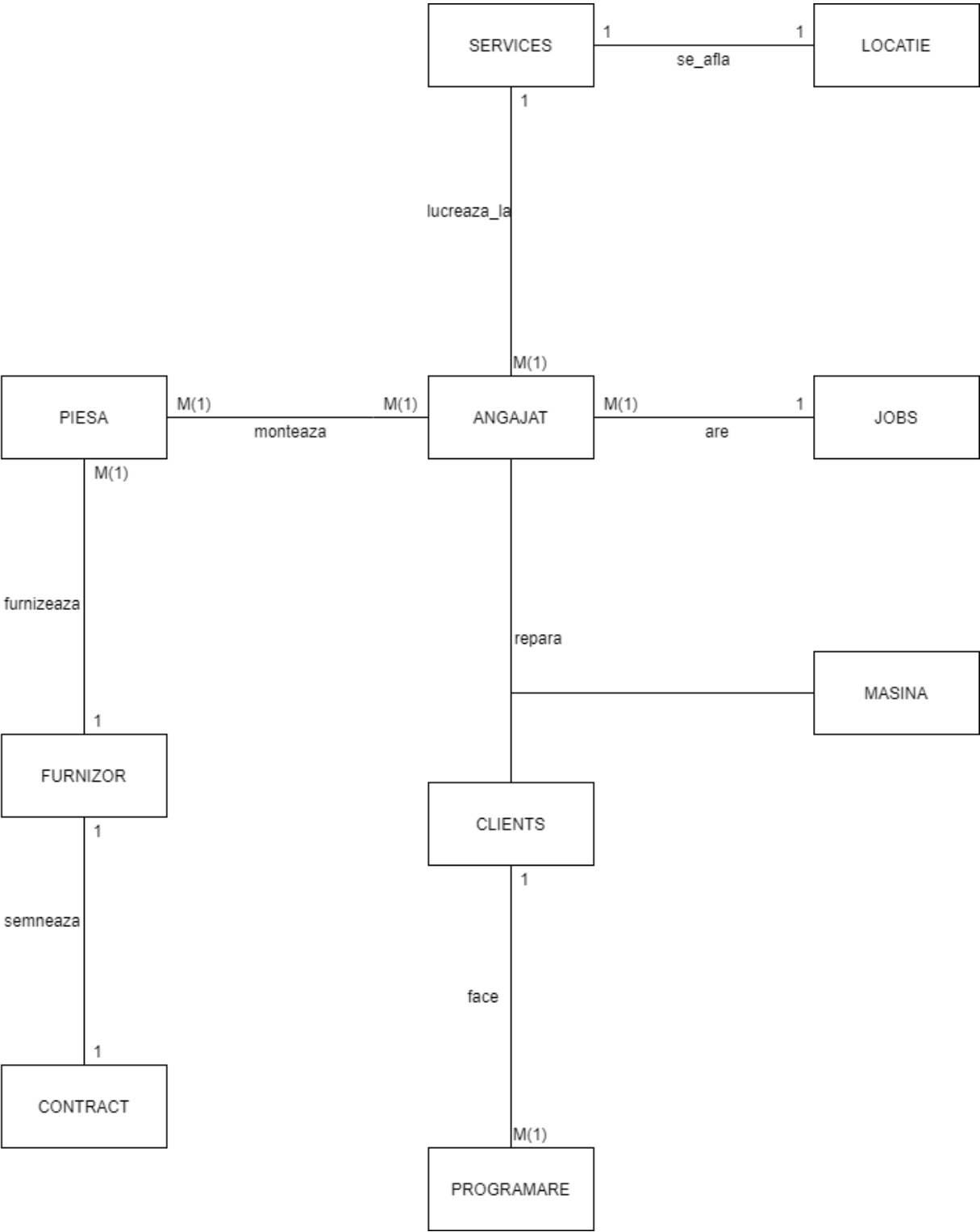
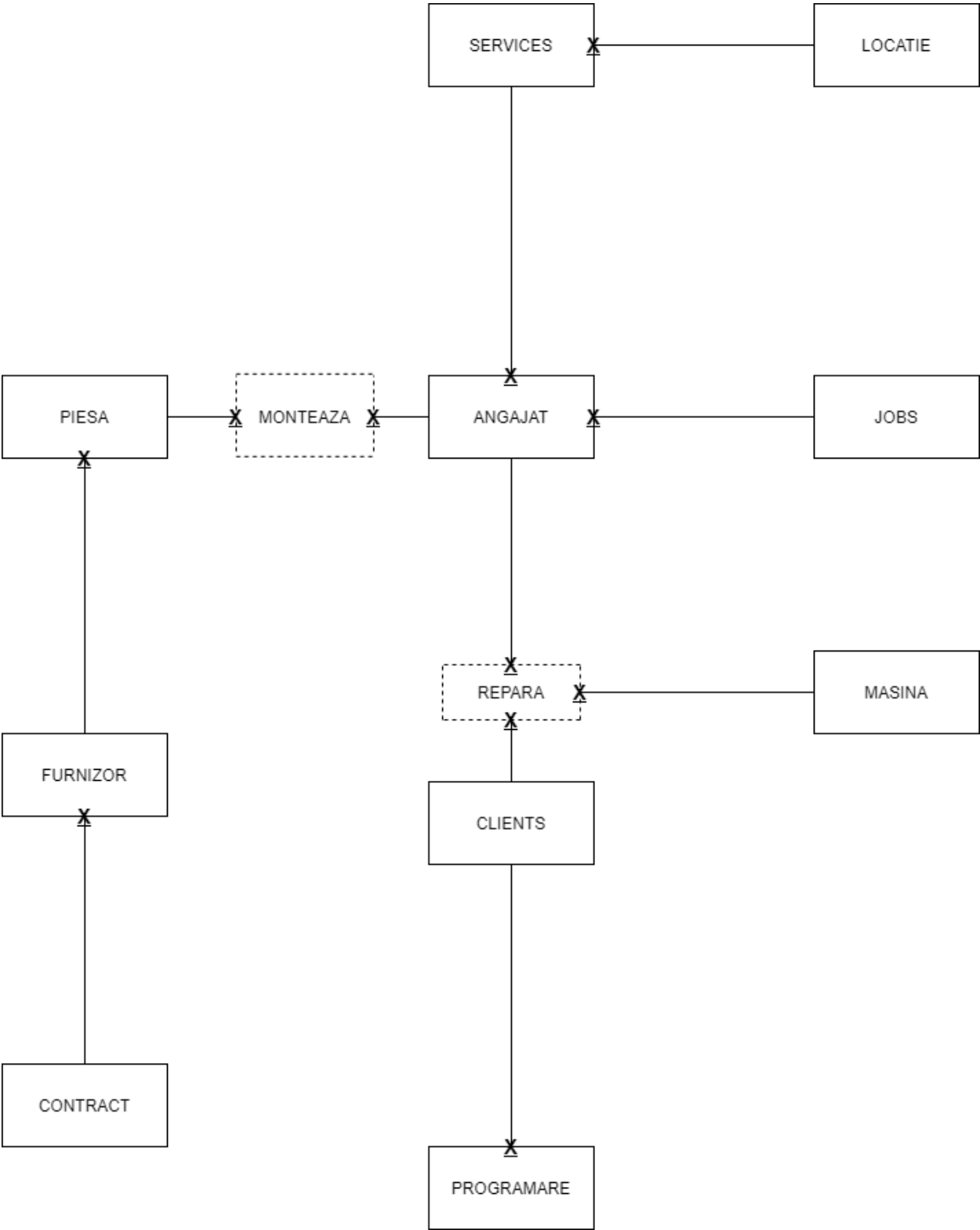


Diagrama conceptuală



Schemele relaționale:

SERVICES (*id_service#*, *nume_service*, *an_infiintare*, *id_locatie*)

LOCATIE (*id_locatie#*, *judet*, *localitate*, *strada*, *numar*, *cod_postal*);

ANGAJAT (*id_angajat#*, *nume_angajat*, *prenume_angajat*, *salariu*, *data_angajarii*, *id_service*, *id_job*);

PIESA (*id_piesa#*, *nume_piesa*, *fabricata_de*, *id_furnizor*);

FURNIZOR (*id_furnizor#*, *nume_furnizor*, *valoare*, *id_contract*);

CONTRACT (*id_contract#*, *data_incepere*, *durata*);

JOBS (*id_job#*, *nume_job*, *salariu_min*, *salariu_max*);

MASINA (*id_masina#*, *marca*, *an_fabricatie*);

CLIENTS (*id_client#*, *nume_client*, *prenume_client*, *numar_telefon*);

PROGRAMARE (*id_programare#*, *data_programare*, *id_client*);

MONTEAZA (*id_angajat#*, *id_piesa#*);

REPARA (*id_angajat#*, *id_masina#*, *id_client#*).

FORMELE NORMALE FN1, FN2, FN3

FORMA NORMALA 1 (FN1)

MONTEAZA (Non-FN1 – exemplu fictiv)

<i>id_angajat#</i>	<i>nume_piesa</i>
1016	‘bujie’, ‘ambreiaj’, ‘cutie de viteze’
1018	‘curea transmisie’, ‘filtru polen’, ‘filtru ulei’

Acest exemplu nu se află în prima formă normală deoarece valorile din coloana *nume_piesa* nu sunt indivizibile, condiție obligatorie pentru FN1.

Transformarea în FN1:

MONTEAZA (FN1)

id_angajat#	nume_piesa
1016	'bujie'
1016	'ambreiaj'
1016	'cutie de viteze'
1018	'curea transmisie'
1018	'filtru polen'
1018	'filtru ulei'

FORMA NORMALA 2 (FN2)

MONTEAZA (Non FN2 – exemplu fictiv)

id_piesa#	id_angajat#	manopera	ore_necesare
100	1016	600	3
100	1015	600	2.5
101	1016	900	1
101	1015	900	1.5
102	1010	300	1.25

Acest exemplu este în FN1 deoarece toate valorile atributelor sunt indivizibile, însă nu este în FN2, pentru că atributul manopera depinde doar de id_piesa#, și nu de întreaga cheie primară compusă, alcătuită din id_piesa# și id_angajat#.

Astfel avem:

{id_piesa#} => {manopera} id_piesa# determină funcțional manopera

{id_piesa#, id_angajat#} => {ore_necesare}.

Transformarea în FN2:

Monteaza_2a:

id_piesa#	id_angajat#	ore_necesare
100	1016	3
100	1015	2.5
101	1016	1
101	1015	1.5
102	1010	1.25

Monteaza_2b:

id_piesa#	manopera
100	600
101	900
102	300

FORMA NORMALĂ 3 (FN3)

MONTEAZA (Non FN3 – exemplu fictiv)

id_piesa#	id_angajat#	manopera	ore_necesare
100	1016	1000	5
100	1015	700	3.5
101	1016	900	4.5
101	1015	1000	5
102	1010	900	4.5

Acest exemplu fictiv se află în FN2, deoarece attributele care nu sunt cheie depind direct de întreaga cheie primară, însă nu se află în FN3, deoarece ele nu depind **direct** de întreaga cheie primară.

{id_piesa#, id_angajat#} => {manopera} cheia primara determina functional manopera

{id_piesa#, id_angajat#} => {manopera} => {ore_necesare}. Astfel, ore_necesare, care nu este un atribut ce participă la cheia primară, nu depinde direct de aceasta, ci prin manopera.

Pentru a aduce relația în FN3 se aplică regula Casey-Delobel. Relația se descompune, prin eliminarea dependențelor funcționale tranzitive, în proiecțiile:

MONTEAZA_3a (id_piesa#, id_angajat#, manopera)

MONTEAZA_3b (manopera, ore_necesare).

Transformarea în FN3:

Monteaza_3a:

id_piesa#	id_angajat#	manopera
100	1016	1000
100	1015	700
101	1016	900
101	1015	1000
102	1010	900

Monteaza_3b:

manopera	ore_necesare
1000	5
700	3.5
900	4.5

CREAREA TABELELOR SI INSERAREA DE VALORI.

Am să las aici codul SQL (de creare, de inserare și de secvențe) alături de print-screenuri care demonstrează faptul că fiecare secvență de cod a fost rulată în SQL. Print-screen-urile vor fi realizate la rularea codului `SELECT* FROM nume_tabel;`

Astfel, pentru tabelul LOCATIE:

```
CREATE TABLE LOCATIE
```

```
(id_locatie number(6),  
judet varchar2(20) constraint judet_nn not null,  
localitate varchar2(30) constraint localitate_nn not null,  
strada varchar2(30) constraint strada_nn not null,  
numar number(3) constraint numar_nn not null,  
cod_postal varchar2(6),  
constraint pk_locatie primary key(id_locatie),  
check(numar > 0)  
);
```

```
CREATE SEQUENCE SEQUENCE_LOCATIE
```

```
INCREMENT BY 1
```

```
START WITH 0
```

```
MINVALUE 0
```

```
MAXVALUE 100
```

```
NOCYCLE;
```

```
INSERT INTO LOCATIE VALUES(SEQUENCE_LOCATIE.NEXTVAL, 'sector 4', 'Bucuresti',
'Giurgiului', '109', '050355');
```

```
INSERT INTO LOCATIE VALUES(SEQUENCE_LOCATIE.NEXTVAL, 'sector 5', 'Bucuresti',
'Ogradei', '4', '050325');
```

```
INSERT INTO LOCATIE VALUES(SEQUENCE_LOCATIE.NEXTVAL, 'Cluj', 'Cluj-Napoca',
'Nicolae Titulescu', '14', '400420');
```

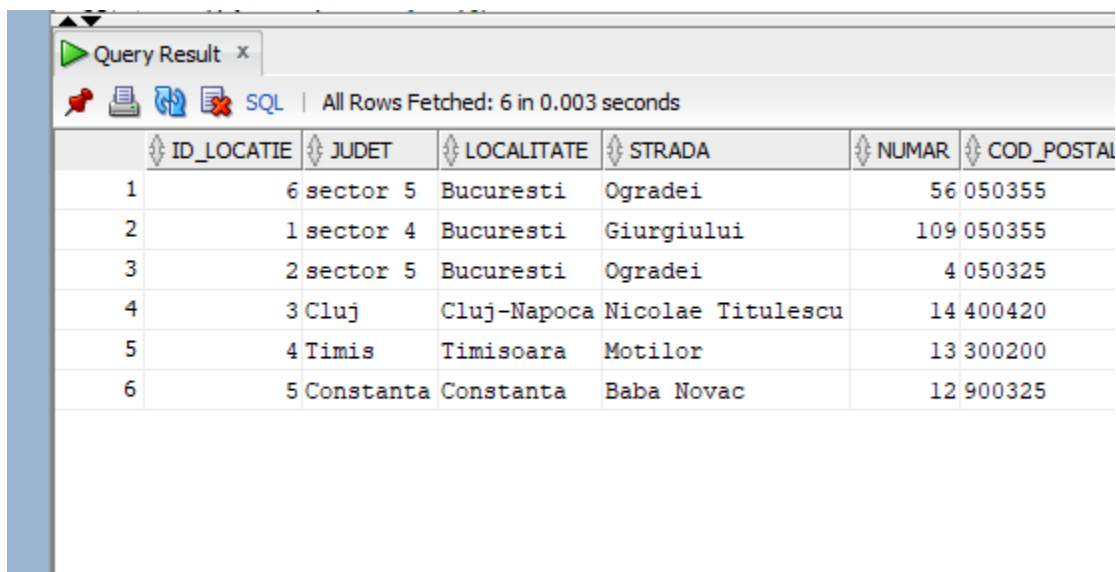
```
INSERT INTO LOCATIE VALUES(SEQUENCE_LOCATIE.NEXTVAL, 'Timis', 'Timisoara',
'Motilor', '13', '300200');
```

```
INSERT INTO LOCATIE VALUES(SEQUENCE_LOCATIE.NEXTVAL, 'Constanta',
'Constanta', 'Baba Novac', '12', '900325');
```

```
INSERT INTO LOCATIE VALUES(SEQUENCE_LOCATIE.NEXTVAL, 'sector 5', 'Bucuresti',
'Ogradei', '56', '050355');
```

```
COMMIT;
```

```
SELECT* FROM LOCATIE;
```



Query Result x

SQL | All Rows Fetched: 6 in 0.003 seconds

	ID_LOCATIE	JUDET	LOCALITATE	STRADA	NUMAR	COD_POSTAL
1		6 sector 5	Bucuresti	Ogradei	56	050355
2		1 sector 4	Bucuresti	Giurgiului	109	050355
3		2 sector 5	Bucuresti	Ogradei	4	050325
4		3 Cluj	Cluj-Napoca	Nicolae Titulescu	14	400420
5		4 Timis	Timisoara	Motilor	13	300200
6		5 Constanta	Constanta	Baba Novac	12	900325

Astfel, pentru tabelul SERVICES:

```
CREATE TABLE SERVICES
```

```
(id_service number(6),
```

```
nume_service varchar2(6) constraint nume_service_nn not null,
```

GESTIUNEA SERVICE-URILOR AUTO DIN ROMÂNIA

```
an_infiintare number(4),
id_locatie number(6),
constraint pk_service primary key(id_service),
constraint fk_service_locatie foreign key(id_locatie) references LOCATIE(id_locatie),
check (an_infiintare > 1900)
);
```

ALTER TABLE services --am modificat dimensiunea numelui service-urilor, deoarece am gresit dimensiunea initiala.

```
MODIFY (nume_service varchar2(40));
```

```
CREATE SEQUENCE SEQUENCE_SERVICE
START WITH 100
INCREMENT BY 1
MINVALUE 0
MAXVALUE 200
NOCYCLE;
```

```
INSERT INTO SERVICES VALUES(SEQUENCE_SERVICE.NEXTVAL, 'Engine-fixers',
'2005', 4);
```

```
INSERT INTO SERVICES VALUES(SEQUENCE_SERVICE.NEXTVAL, 'Crazy-Wheel',
'1998', 5);
```

```
INSERT INTO SERVICES VALUES(SEQUENCE_SERVICE.NEXTVAL, 'ReadytoGo', '2015',
2);
```

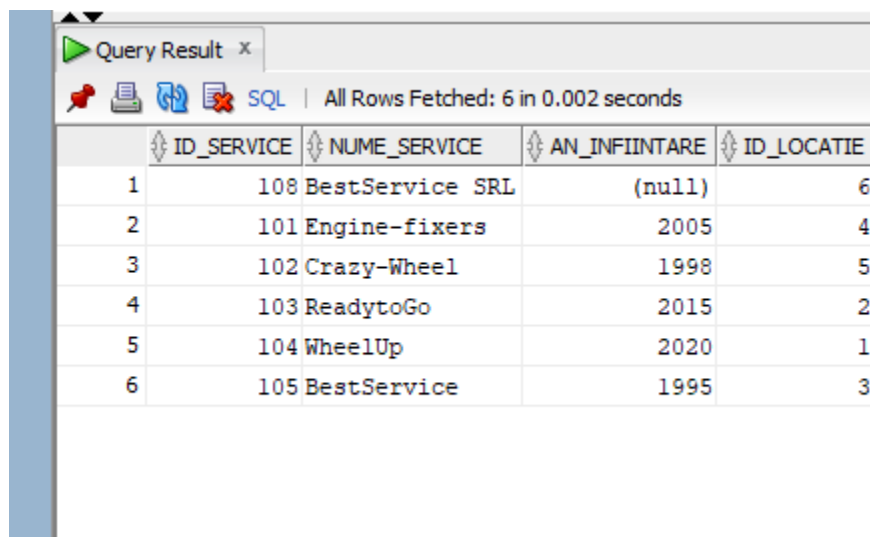
```
INSERT INTO SERVICES VALUES(SEQUENCE_SERVICE.NEXTVAL, 'WheelUp', '2020',
1);
```

```
INSERT INTO SERVICES VALUES(SEQUENCE_SERVICE.NEXTVAL, 'BestService', '1995',
3);
```

```
INSERT INTO SERVICES VALUES(SEQUENCE_SERVICE.NEXTVAL, 'BestService
SRL',NULL, 6);
```

```
COMMIT;
```

SELECT* FROM SERVICES;



Query Result x

All Rows Fetched: 6 in 0.002 seconds

	ID_SERVICE	NUME_SERVICE	AN_INFIINTARE	ID_LOCATIE
1	108	BestService SRL	(null)	6
2	101	Engine-fixers	2005	4
3	102	Crazy-Wheel	1998	5
4	103	ReadytoGo	2015	2
5	104	WheelUp	2020	1
6	105	BestService	1995	3

Astfel, pentru tabelul JOBS:

CREATE TABLE JOBS

```
(id_job number(6),
nume_job varchar2(30) constraint nume_job_nn not null,
salariu_max number(6) constraint salariu_max_nn not null,
salariu_min number(6) constraint salariu_min_nn not null,
constraint pk_jobs primary key(id_job)
);
```

INSERT INTO JOBS VALUES(500, 'Mecanic', '30000', '9000');

INSERT INTO JOBS VALUES(501, 'Tinichigiu', '20000', '5000');

INSERT INTO JOBS VALUES(502, 'Electrician auto', '25000', '6500');

INSERT INTO JOBS VALUES(503, 'Contabil', '22500', '9300');

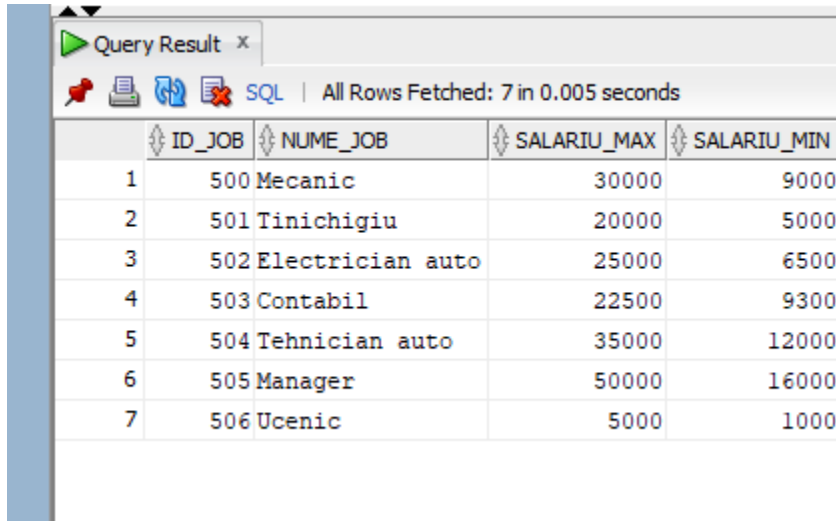
INSERT INTO JOBS VALUES(504, 'Tehnician auto', '35000', '12000');

INSERT INTO JOBS VALUES(505, 'Manager', '50000', '16000');

INSERT INTO JOBS VALUES(506, 'Ucenic', '5000', '1000');

COMMIT;

SELECT* FROM JOBS;



The screenshot shows a 'Query Result' window with a table containing 7 rows and 5 columns. The columns are ID_JOB, NUME_JOB, SALARIU_MAX, and SALARIU_MIN. The data is as follows:

	ID_JOB	NUME_JOB	SALARIU_MAX	SALARIU_MIN
1	500	Mecanic	30000	9000
2	501	Tinichigiu	20000	5000
3	502	Electrician auto	25000	6500
4	503	Contabil	22500	9300
5	504	Tehnician auto	35000	12000
6	505	Manager	50000	16000
7	506	Ucenic	5000	1000

Astfel, pentru tabelul ANGAJAT:

CREATE TABLE ANGAJAT

```
(id_angajat number(6),  
  nume_angajat varchar2(30) constraint nume_angajat_nn not null,  
  prenume_angajat varchar2(30) constraint prenume_angajat_nn not null,  
  salariu number(6) constraint salariu_nn not null,  
  data_angajarii date default sysdate,  
  id_service number(6),  
  id_job number(6),  
  constraint pk_angajat primary key(id_angajat),  
  constraint fk_angajat_service foreign key (id_service) references SERVICES(id_service),  
  constraint fk_angajat_job foreign key(id_job) references JOBS(id_job)  
);
```



```
CREATE SEQUENCE SEQUENCE_ANGAJAT
```

```
START WITH 1003
```

```
INCREMENT BY 1
```

```
MAXVALUE 2000
```

```
NOCYCLE;
```

```
INSERT INTO ANGAJAT VALUES(SEQUENCE_ANGAJAT.NEXTVAL, 'Vasile', 'Ionel',  
'12000', TO_DATE('23/11/2009', 'DD/MM/YYYY'), '102', '500');
```

```
INSERT INTO ANGAJAT VALUES(SEQUENCE_ANGAJAT.NEXTVAL, 'Ionescu', 'Gigel',  
'10000', TO_DATE('28/10/2003', 'DD/MM/YYYY'), '102', '502');
```

```
INSERT INTO ANGAJAT VALUES(SEQUENCE_ANGAJAT.NEXTVAL, 'Vasilescu', 'Ion',  
'11550', TO_DATE('11/02/1999', 'DD/MM/YYYY'), '102', '503');
```

```
INSERT INTO ANGAJAT VALUES(SEQUENCE_ANGAJAT.NEXTVAL, 'Popescu', 'Marin',  
'16000', TO_DATE('19/07/2019', 'DD/MM/YYYY'), '101', '500');
```

```
INSERT INTO ANGAJAT VALUES(SEQUENCE_ANGAJAT.NEXTVAL, 'Iordache', 'Adrian',  
'6000', TO_DATE('23/05/2017', 'DD/MM/YYYY'), '101', '501');
```

```
INSERT INTO ANGAJAT VALUES(SEQUENCE_ANGAJAT.NEXTVAL, 'Enescu', 'Alin',  
'42500', TO_DATE('05/03/2007', 'DD/MM/YYYY'), '101', '505');
```

```
INSERT INTO ANGAJAT VALUES(SEQUENCE_ANGAJAT.NEXTVAL, 'Deculescu',  
'Emanuel', '10000', TO_DATE('26/04/2016', 'DD/MM/YYYY'), '103', '503');
```

```
INSERT INTO ANGAJAT VALUES(SEQUENCE_ANGAJAT.NEXTVAL, 'Mircescu', 'Ovidiu',  
'95000', TO_DATE('28/08/2016', 'DD/MM/YYYY'), '103', '500');
```

```
INSERT INTO ANGAJAT VALUES(SEQUENCE_ANGAJAT.NEXTVAL, 'Sandu', 'Mircea',  
'7000', TO_DATE('21/12/2019', 'DD/MM/YYYY'), '103', '501');
```

```
INSERT INTO ANGAJAT VALUES(SEQUENCE_ANGAJAT.NEXTVAL, 'Patrascu', 'Dorel',  
'18550', TO_DATE('13/04/2021', 'DD/MM/YYYY'), '104', '504');
```

```
INSERT INTO ANGAJAT VALUES(SEQUENCE_ANGAJAT.NEXTVAL, 'Davidescu', 'Filip',  
'10800', TO_DATE('15/02/2021', 'DD/MM/YYYY'), '104', '503');
```

```
INSERT INTO ANGAJAT VALUES(SEQUENCE_ANGAJAT.NEXTVAL, 'Isarescu', 'Iulian',  
'25000', TO_DATE('29/11/2020', 'DD/MM/YYYY'), '104', '505');
```

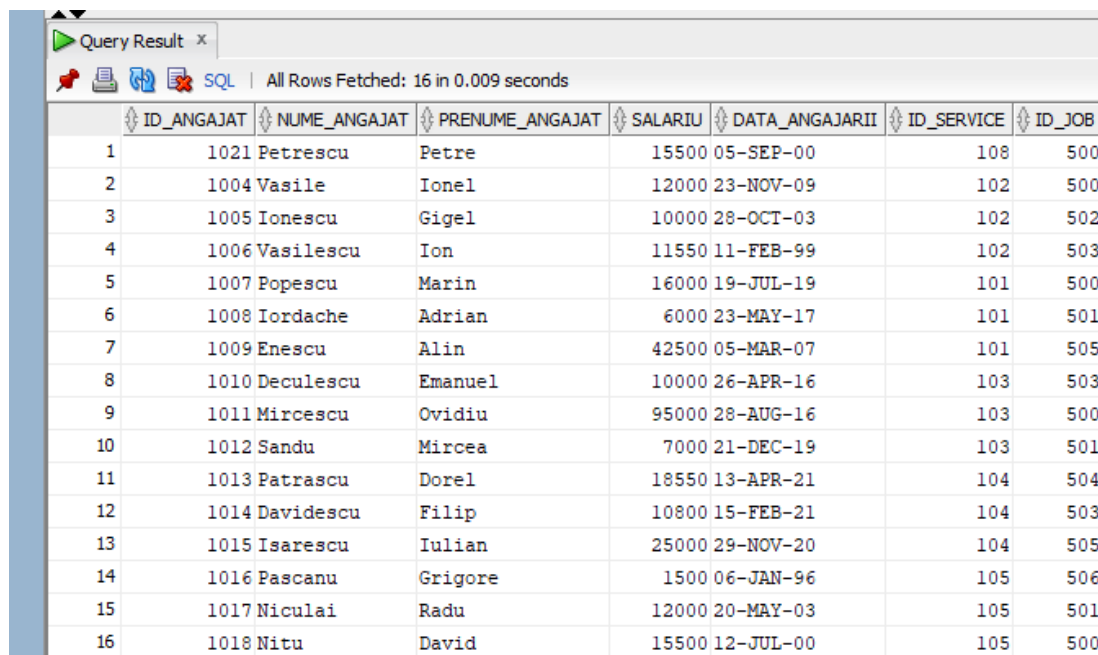
```
INSERT INTO ANGAJAT VALUES(SEQUENCE_ANGAJAT.NEXTVAL, 'Pascanu', 'Grigore',  
'1500', TO_DATE('06/01/1996', 'DD/MM/YYYY'), '105', '506');
```

```
INSERT INTO ANGAJAT VALUES(SEQUENCE_ANGAJAT.NEXTVAL, 'Niculai', 'Radu',
'12000', TO_DATE('20/05/2003', 'DD/MM/YYYY'), '105', '501');
```

```
INSERT INTO ANGAJAT VALUES(SEQUENCE_ANGAJAT.NEXTVAL, 'Petrescu', 'Petre',
'15500', TO_DATE('5/09/2000', 'DD/MM/YYYY'), '108', '500');
```

```
COMMIT;
```

```
SELECT* FROM ANGAJAT;
```



Query Result x

All Rows Fetched: 16 in 0.009 seconds

	ID_ANGAJAT	NUME_ANGAJAT	PRENUME_ANGAJAT	SALARIU	DATA_ANGAJARII	ID_SERVICE	ID_JOB
1	1021	Petrescu	Petre	15500	05-SEP-00	108	500
2	1004	Vasile	Ionel	12000	23-NOV-09	102	500
3	1005	Ionescu	Gigel	10000	28-OCT-03	102	502
4	1006	Vasilescu	Ion	11550	11-FEB-99	102	503
5	1007	Popescu	Marin	16000	19-JUL-19	101	500
6	1008	Iordache	Adrian	6000	23-MAY-17	101	501
7	1009	Enescu	Alin	42500	05-MAR-07	101	505
8	1010	Deculescu	Emanuel	10000	26-APR-16	103	503
9	1011	Mircescu	Ovidiu	95000	28-AUG-16	103	500
10	1012	Sandu	Mircea	7000	21-DEC-19	103	501
11	1013	Patrascu	Dorel	18550	13-APR-21	104	504
12	1014	Davidescu	Filip	10800	15-FEB-21	104	503
13	1015	Isarescu	Iulian	25000	29-NOV-20	104	505
14	1016	Pascanu	Grigore	1500	06-JAN-96	105	506
15	1017	Niculai	Radu	12000	20-MAY-03	105	501
16	1018	Nitu	David	15500	12-JUL-00	105	500

Astfel, pentru tabelul CLIENTS:

```
CREATE TABLE CLIENTS
```

```
(id_client number(6),
```

```
nume_client varchar2(30) constraint nume_client_nn not null,
```

```
prenume_client varchar2(30),
```

```
numar_telefon varchar2(13),
```

```
constraint pk_client primary key(id_client)
```

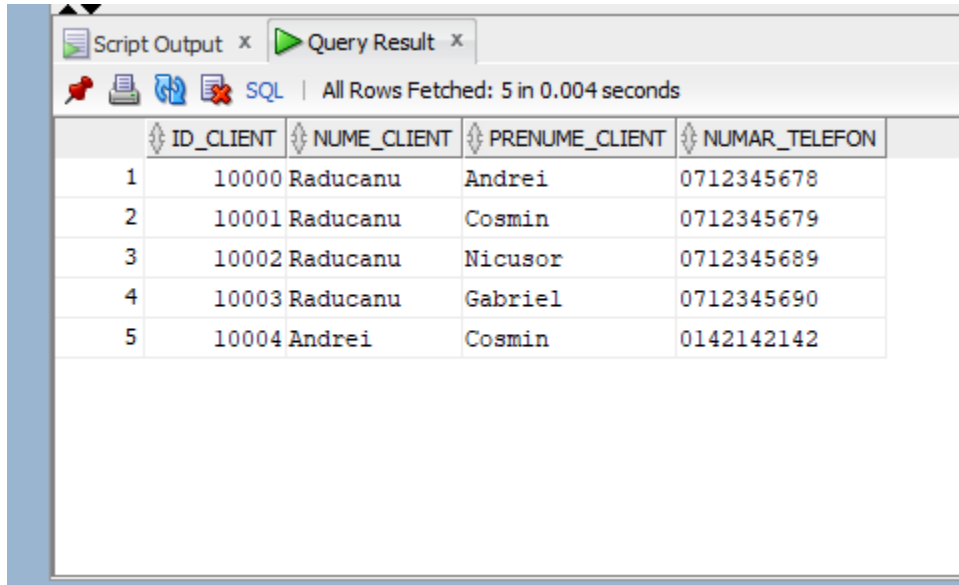
```
);
```

```
INSERT INTO CLIENTS VALUES('10000', 'Raducanu', 'Andrei', '0712345678');
```

```
INSERT INTO CLIENTS VALUES('10001', 'Raducanu', 'Cosmin', '0712345679');
```

```
INSERT INTO CLIENTS VALUES('10002', 'Raducanu', 'Nicusor', '0712345689');
INSERT INTO CLIENTS VALUES('10003', 'Raducanu', 'Gabriel', '0712345690');
INSERT INTO CLIENTS VALUES('10004', 'Andrei', 'Cosmin', '0142142142');
COMMIT;
```

```
SELECT* FROM CLIENTS;
```



The screenshot shows a SQL query result window with the following data:

	ID_CLIENT	NUME_CLIENT	PRENUME_CLIENT	NUMAR_TELEFON
1	10000	Raducanu	Andrei	0712345678
2	10001	Raducanu	Cosmin	0712345679
3	10002	Raducanu	Nicusor	0712345689
4	10003	Raducanu	Gabriel	0712345690
5	10004	Andrei	Cosmin	0142142142

Astfel, pentru tabelul MASINA:

```
CREATE TABLE MASINA
```

```
(
```

```
id_masina number(6),
```

```
marca varchar(20) constraint marca_nn not null,
```

```
an_fabricatie number(4) constraint an_fabricatie_nn not null,
```

```
constraint pk_masina primary key(id_masina),
```

```
check(an_fabricatie>1960)
```

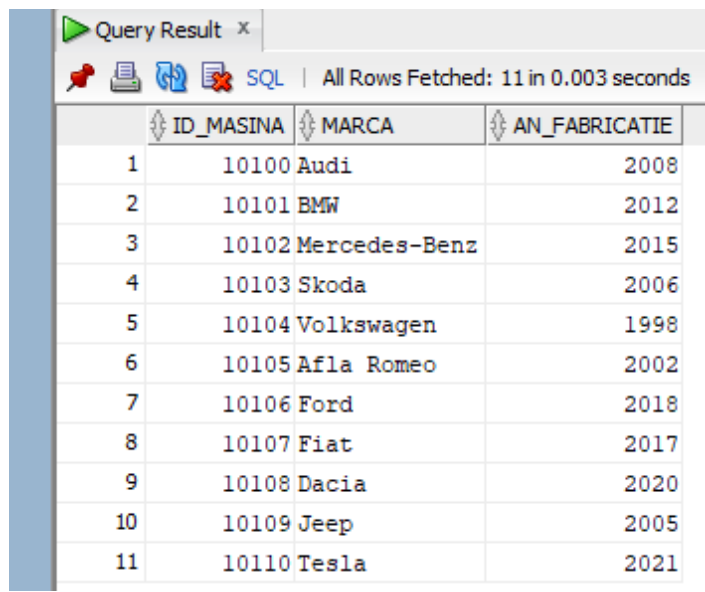
```
);
```

```
INSERT INTO MASINA VALUES('10100', 'Audi', '2008');
```

```
INSERT INTO MASINA VALUES('10101', 'BMW', '2012');
```

```
INSERT INTO MASINA VALUES('10102', 'Mercedes-Benz', '2015');
INSERT INTO MASINA VALUES('10103', 'Skoda', '2006');
INSERT INTO MASINA VALUES('10104', 'Volkswagen', '1998');
INSERT INTO MASINA VALUES('10105', 'Afla Romeo', '2002');
INSERT INTO MASINA VALUES('10106', 'Ford', '2018');
INSERT INTO MASINA VALUES('10107', 'Fiat', '2017');
INSERT INTO MASINA VALUES('10108', 'Dacia', '2020');
INSERT INTO MASINA VALUES('10109', 'Jeep', '2005');
INSERT INTO MASINA VALUES('10110', 'Tesla', '2021');
COMMIT;
```

```
SELECT* FROM MASINA;
```



The screenshot shows a 'Query Result' window with 11 rows of data. The columns are ID_MASINA, MARCA, and AN_FABRICATIE. The data is as follows:

	ID_MASINA	MARCA	AN_FABRICATIE
1	10100	Audi	2008
2	10101	BMW	2012
3	10102	Mercedes-Benz	2015
4	10103	Skoda	2006
5	10104	Volkswagen	1998
6	10105	Afla Romeo	2002
7	10106	Ford	2018
8	10107	Fiat	2017
9	10108	Dacia	2020
10	10109	Jeep	2005
11	10110	Tesla	2021

Astfel, pentru tabelul PROGRAMARE:

```
CREATE TABLE PROGRAMARE
```

```
(id_programare number(6),
```

```
data_programare date default sysdate,
```

```
id_client number(6),
```

```
constraint pk_programare primary key(id_programare),
constraint fk_programare_client foreign key(id_client) references CLIENTS(id_client)
);
```

```
INSERT INTO PROGRAMARE VALUES('100000', TO_DATE('31/05/2021',
'DD/MM/YYYY'), '10000');
```

```
INSERT INTO PROGRAMARE VALUES('100001', TO_DATE('21/06/2021',
'DD/MM/YYYY'), '10001');
```

```
INSERT INTO PROGRAMARE VALUES('100002', TO_DATE('17/06/2021',
'DD/MM/YYYY'), '10000');
```

```
INSERT INTO PROGRAMARE VALUES('100003', TO_DATE('15/07/2021',
'DD/MM/YYYY'), '10002');
```

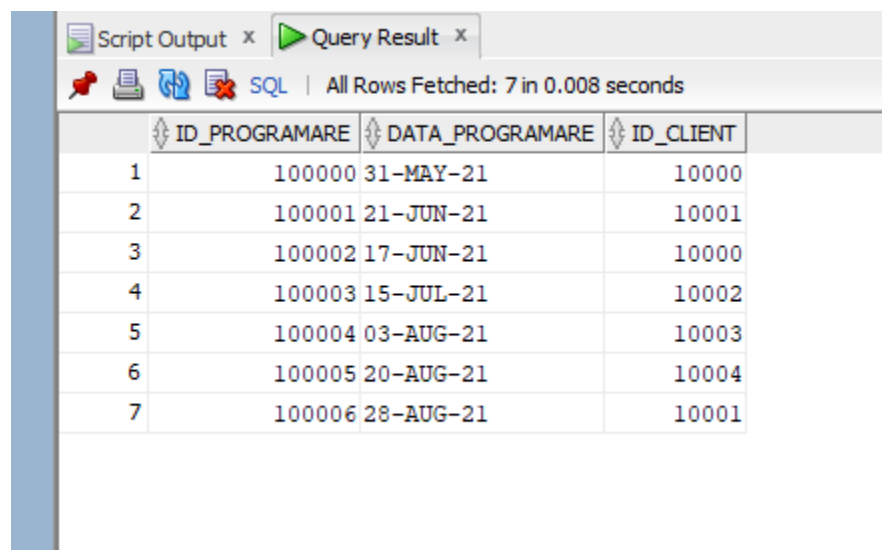
```
INSERT INTO PROGRAMARE VALUES('100004', TO_DATE('03/08/2021',
'DD/MM/YYYY'), '10003');
```

```
INSERT INTO PROGRAMARE VALUES('100005', TO_DATE('20/08/2021',
'DD/MM/YYYY'), '10004');
```

```
INSERT INTO PROGRAMARE VALUES('100006', TO_DATE('28/08/2021',
'DD/MM/YYYY'), '10001');
```

```
COMMIT;
```

```
SELECT* FROM PROGRAMARE;
```



	ID_PROGRAMARE	DATA_PROGRAMARE	ID_CLIENT
1	100000	31-MAY-21	10000
2	100001	21-JUN-21	10001
3	100002	17-JUN-21	10000
4	100003	15-JUL-21	10002
5	100004	03-AUG-21	10003
6	100005	20-AUG-21	10004
7	100006	28-AUG-21	10001

Astfel, pentru tabelul CONTRACT:

CREATE TABLE CONTRACT

```
(id_contract number(6),  
data_incepere date constraint data_incepere_nn not null,  
durata number(3, 3) constraint durata_nn not null,  
constraint pk_contract primary key(id_contract)  
);
```

ALTER TABLE CONTRACT --am modificat dimensiunea atributului durata, deoarece era greșită
MODIFY DURATA NUMBER(6, 2);

INSERT INTO CONTRACT VALUES('20000', TO_DATE('22/07/2019', 'DD/MM/YYYY'), '3');

INSERT INTO CONTRACT VALUES('20001', TO_DATE('18/09/2018', 'DD/MM/YYYY'), '3.5');

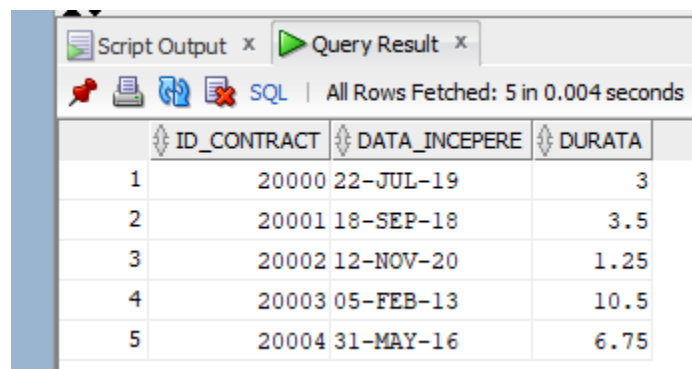
INSERT INTO CONTRACT VALUES('20002', TO_DATE('12/11/2020', 'DD/MM/YYYY'), '1.25');

INSERT INTO CONTRACT VALUES('20003', TO_DATE('05/02/2013', 'DD/MM/YYYY'), '10.5');

INSERT INTO CONTRACT VALUES('20004', TO_DATE('31/05/2016', 'DD/MM/YYYY'), '6.75');

COMMIT;

SELECT* FROM CONTRACT;



The screenshot shows a database application window with two tabs: 'Script Output' and 'Query Result'. The 'Query Result' tab is active, displaying the results of a SELECT query. The window title bar includes icons for saving, refreshing, and deleting, along with the text 'SQL | All Rows Fetched: 5 in 0.004 seconds'. The data is presented in a table with three columns: 'ID_CONTRACT', 'DATA_INCEPERE', and 'DURATA'. The rows are numbered 1 through 5 in the first column.

	ID_CONTRACT	DATA_INCEPERE	DURATA
1	20000	22-JUL-19	3
2	20001	18-SEP-18	3.5
3	20002	12-NOV-20	1.25
4	20003	05-FEB-13	10.5
5	20004	31-MAY-16	6.75

Astfel, pentru tabelul FURNIZOR:

CREATE TABLE FURNIZOR

```
(id_furnizor number(6),  
  nume_furnizor varchar2(40) constraint nume_furnizor_nn not null,  
  valoare number(8) constraint valoare_nn not null,  
  id_contract number(6),  
  constraint pk_furnizor primary key(id_furnizor),  
  constraint fk_furnizor_contract foreign key(id_contract) references CONTRACT(id_contract)  
);
```

INSERT INTO FURNIZOR VALUES('30000', 'Auto Rebel', '1000000', '20000');

INSERT INTO FURNIZOR VALUES('30001', 'Lipi Impex 2110', '1800000', '20004');

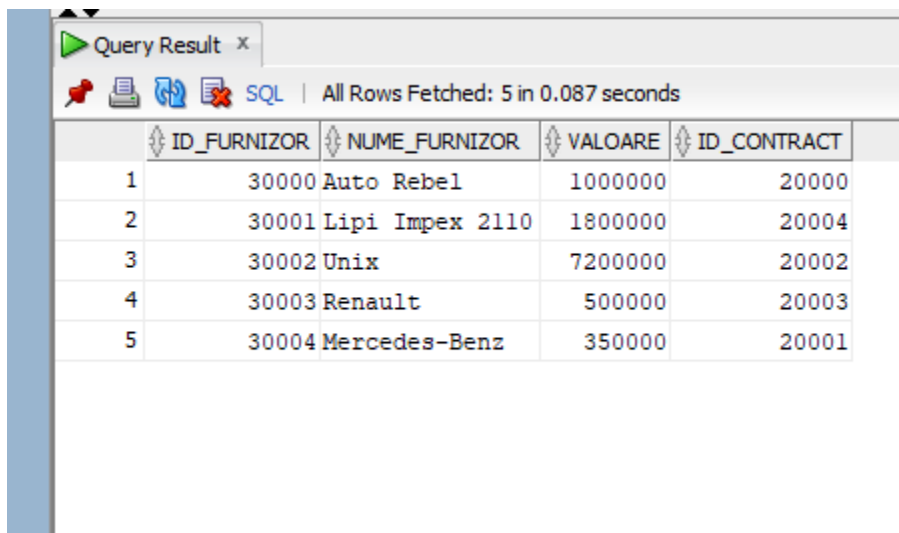
INSERT INTO FURNIZOR VALUES('30002', 'Unix', '7200000', '20002');

INSERT INTO FURNIZOR VALUES('30003', 'Renault', '500000', '20003');

INSERT INTO FURNIZOR VALUES('30004', 'Mercedes-Benz', '350000', '20001');

COMMIT;

SELECT* FROM FURNIZOR;



The screenshot shows a 'Query Result' window with the following data:

	ID_FURNIZOR	NUME_FURNIZOR	VALOARE	ID_CONTRACT
1	30000	Auto Rebel	1000000	20000
2	30001	Lipi Impex 2110	1800000	20004
3	30002	Unix	7200000	20002
4	30003	Renault	500000	20003
5	30004	Mercedes-Benz	350000	20001

Astfel, pentru tabelul PIESA:

CREATE TABLE PIESA

```
(id_piesa number(6),  
 nume_piesa varchar2(35) constraint nume_piesa_nn not null,  
 fabricata_de varchar2(40) constraint fabricata_de_nn not null,  
 id_furnizor number(6),  
 constraint pk_piesa primary key(id_piesa),  
 constraint fk_piesa_furnizor foreign key(id_furnizor) references FURNIZOR(id_furnizor)  
);
```

INSERT INTO PIESA VALUES('40000', 'Bujie', 'Renault', '30004');

INSERT INTO PIESA VALUES('40001', 'Ambreiaj', 'Mercedes-Benz', '30002');

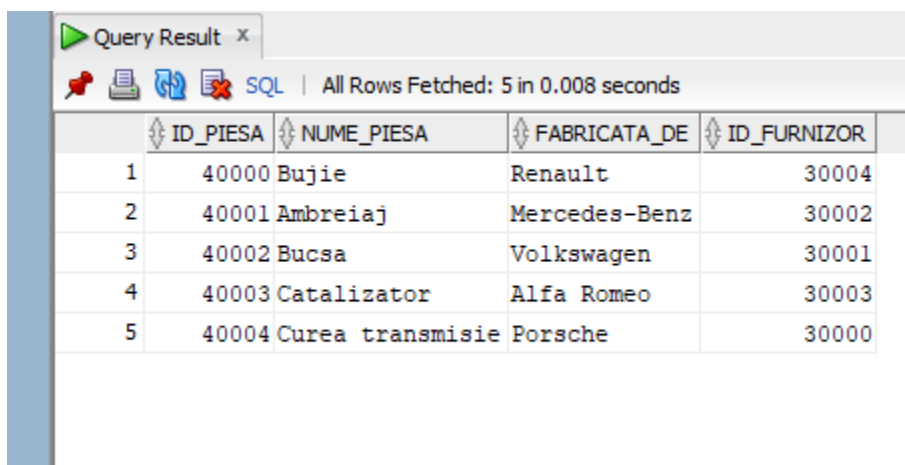
INSERT INTO PIESA VALUES('40002', 'Bucsa', 'Volkswagen', '30001');

INSERT INTO PIESA VALUES('40003', 'Catalizator', 'Alfa Romeo', '30003');

INSERT INTO PIESA VALUES('40004', 'Curea transmisie', 'Porsche', '30000');

COMMIT;

SELECT* FROM PIESA;



Query Result x

SQL | All Rows Fetched: 5 in 0.008 seconds

	ID_PIESA	NUME_PIESA	FABRICATA_DE	ID_FURNIZOR
1	40000	Bujie	Renault	30004
2	40001	Ambreiaj	Mercedes-Benz	30002
3	40002	Bucsa	Volkswagen	30001
4	40003	Catalizator	Alfa Romeo	30003
5	40004	Curea transmisie	Porsche	30000

Astfel, pentru tabelul MONTEAZA:

CREATE TABLE MONTEAZA

```
(id_angajat number(6),
id_piesa number(6),
constraint pk_monteaza primary key(id_angajat, id_piesa),
constraint fk_monteaza_angajat foreign key(id_angajat) references ANGAJAT(id_angajat),
constraint fk_monteaza_piesa foreign key(id_piesa) references PIESA(id_piesa)
);
```

INSERT INTO MONTEAZA VALUES('1004', '40000');

INSERT INTO MONTEAZA VALUES('1004', '40003');

INSERT INTO MONTEAZA VALUES('1005', '40002');

INSERT INTO MONTEAZA VALUES('1005', '40001');

INSERT INTO MONTEAZA VALUES('1006', '40002');

INSERT INTO MONTEAZA VALUES('1007', '40001');

INSERT INTO MONTEAZA VALUES('1009', '40003');

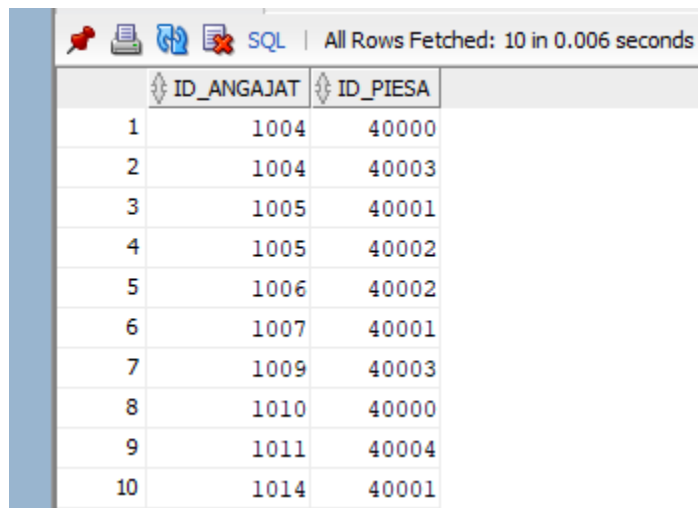
INSERT INTO MONTEAZA VALUES('1011', '40004');

INSERT INTO MONTEAZA VALUES('1010', '40000');

INSERT INTO MONTEAZA VALUES('1014', '40001');

COMMIT;

SELECT* FROM MONTEAZA;



	ID_ANGAJAT	ID_PIESA
1	1004	40000
2	1004	40003
3	1005	40001
4	1005	40002
5	1006	40002
6	1007	40001
7	1009	40003
8	1010	40000
9	1011	40004
10	1014	40001

Astfel, pentru tabelul REPARA:

CREATE TABLE REPARA

```
(id_angajat number(6),  
id_masina number(6),  
id_client number(6),  
constraint pk_repara primary key(id_angajat, id_masina, id_client),  
constraint fk_repara_angajat foreign key(id_angajat) references ANGAJAT(id_angajat),  
constraint fk_repara_client foreign key(id_client) references CLIENTS(id_client),  
constraint fk_repara_masina foreign key(id_masina) references MASINA(id_masina)  
);
```

INSERT INTO REPARA VALUES('1005', '10100', '10003');

INSERT INTO REPARA VALUES('1004', '10101', '10002');

INSERT INTO REPARA VALUES('1006', '10102', '10000');

INSERT INTO REPARA VALUES('1007', '10103', '10001');

INSERT INTO REPARA VALUES('1009', '10104', '10004');

INSERT INTO REPARA VALUES('1010', '10105', '10002');

INSERT INTO REPARA VALUES('1011', '10106', '10001');

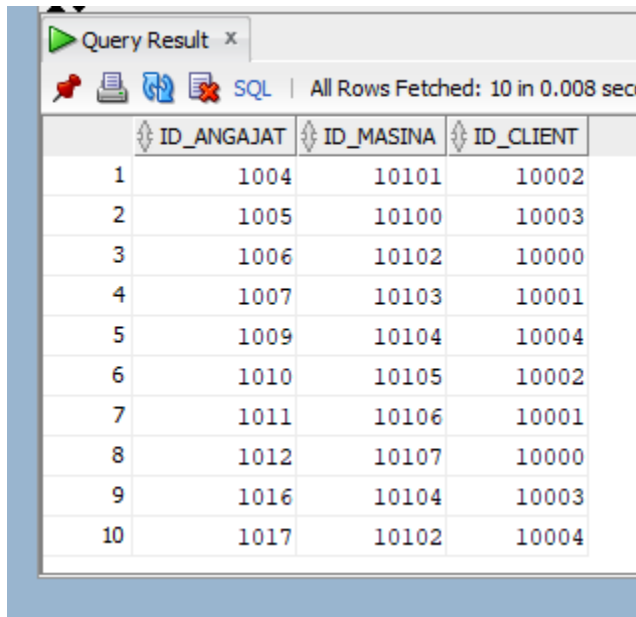
INSERT INTO REPARA VALUES('1012', '10107', '10000');

INSERT INTO REPARA VALUES('1016', '10104', '10003');

INSERT INTO REPARA VALUES('1017', '10102', '10004');

COMMIT;

SELECT* FROM REPARA;



The screenshot shows a 'Query Result' window with a toolbar at the top containing icons for saving, refreshing, and other database actions. Below the toolbar, it indicates 'All Rows Fetched: 10 in 0.008 sec'. The main area displays a table with 4 columns: ID_ANGAJAT, ID_MASINA, ID_CLIENT, and an unlabeled column. The data is as follows:

	ID_ANGAJAT	ID_MASINA	ID_CLIENT	
1	1004	10101	10002	
2	1005	10100	10003	
3	1006	10102	10000	
4	1007	10103	10001	
5	1009	10104	10004	
6	1010	10105	10002	
7	1011	10106	10001	
8	1012	10107	10000	
9	1016	10104	10003	
10	1017	10102	10004	

Ex. 11 - Cereri SQL

--1. Scrieti o cerere SQL in care sa se afiseze codul, numele si prenumele, data angajarii a
-- angajatilor care lucreaza ca mecanic sau lucreaza in bucuresti si sunt angajati dupa data
-- de 14 septembrie 2020. Sa se afiseze si id_ul service-ului in care lucreaza, anul infiintarii
--acestuia(0 daca e null), numele, id_locatiei in care se afla, localitatea, id_ul jobului
--angajatilor si numele acestuia.

Pentru această cerere SQL am utilizat:

- NVL pentru a nu afișa null
- Join-uri pe 4 tabele
- Filtrare la nivel de linii prin clauza WHERE
- Funcțiile pe șiruri de caractere UPPER și LOWER
- Funcția pe date calendaristice ADD_MONTHS
- Ordonare a datelor (ORDER BY)

```

SELECT id_angajat, nume_angajat, prenume_angajat, data_angajarii, s.id_service,
NVL(an_infiintare, 0), nume_service, l.id_locatie, localitate, j.id_job, nume_job

FROM angajat a JOIN services s ON(a.id_service = s.id_service)

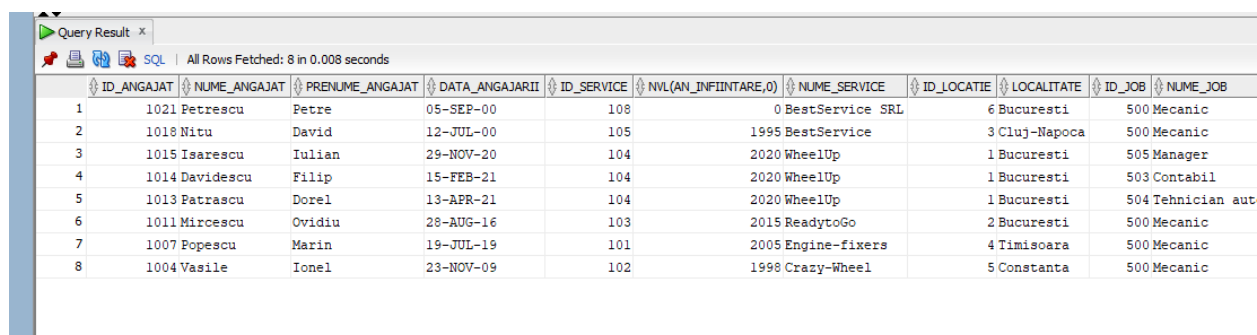
JOIN locatie l ON(s.id_locatie = l.id_locatie)

JOIN jobs j ON(a.id_job = j.id_job)

WHERE UPPER(nume_job) LIKE '%MECANIC%' OR (LOWER(localitate) LIKE
'%bucuresti%' AND data_angajarii > ADD_MONTHS('14-MAR-2020', 6))

ORDER BY id_angajat desc;

```



Query Result x | All Rows Fetched: 8 in 0.008 seconds

ID_ANGAJAT	NUME_ANGAJAT	PRENUME_ANGAJAT	DATA_ANGAJARII	ID_SERVICE	NVL(AN_INFIINTARE,0)	NUME_SERVICE	ID_LOCATIE	LOCALITATE	ID_JOB	NUME_JOB
1	1021 Petrescu	Petre	05-SEP-00	108		0 BestService SRL	6	Bucuresti	500	Mecanic
2	1018 Nitu	David	12-JUL-00	105		1995 BestService	3	Cluj-Napoca	500	Mecanic
3	1015 Isarescu	Iulian	29-NOV-20	104		2020 WheelUp	1	Bucuresti	505	Manager
4	1014 Davidescu	Filip	15-FEB-21	104		2020 WheelUp	1	Bucuresti	503	Contabil
5	1013 Patrascu	Dorel	13-APR-21	104		2020 WheelUp	1	Bucuresti	504	Tehnician aut
6	1011 Mircescu	Ovidiu	28-AUG-16	103		2015 ReadytoGo	2	Bucuresti	500	Mecanic
7	1007 Popescu	Marin	19-JUL-19	101		2005 Engine-fixers	4	Timisoara	500	Mecanic
8	1004 Vasile	Ionel	23-NOV-09	102		1998 Crazy-Wheel	5	Constanta	500	Mecanic

--2. Scrieti o cerere SQL in care sa se afiseze codul, numele si prenumele concatenate ale angajatilor --care lucreaza in service-ul ce se afla in locatia cu codul maxim.

Pentru aceasta cerere SQL am folosit:

- subcereri necorelate/nesincronizate pe 3 tabele
- functia pe şiruri de caractere CONCAT

```

SELECT id_angajat, CONCAT(nume_angajat||' ', prenume_angajat)"Nume si Prenume"

FROM angajat a

WHERE a.id_service IN

(SELECT id_service

FROM services s

WHERE s.id_locatie =

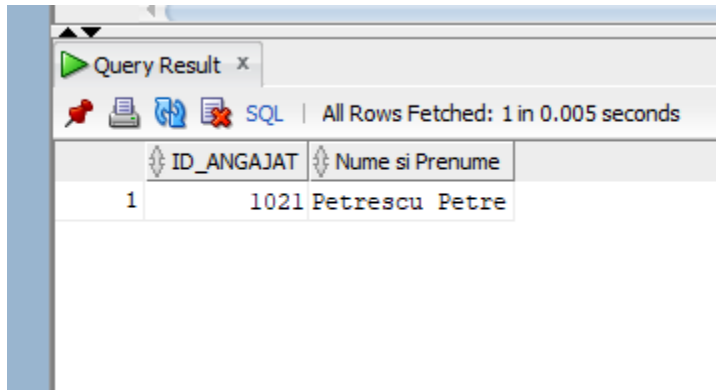
(SELECT MAX(id_locatie)

FROM locatie l

)

);

```



Query Result x

SQL | All Rows Fetched: 1 in 0.005 seconds

	ID_ANGAJAT	Nume si Prenume
1	1021	Petrescu Petre

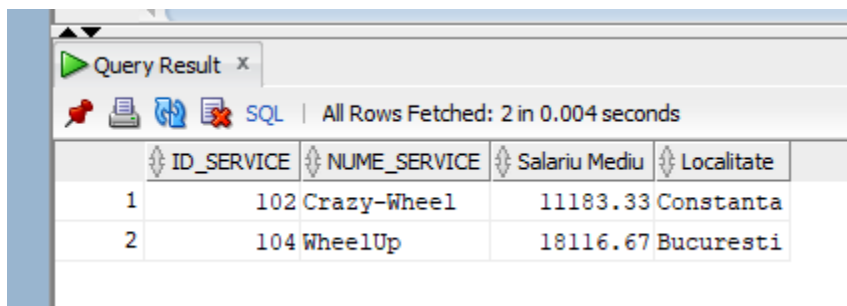
--3. Scrieti o cerere SQL care afiseaza codul si numele service-urilor ale caror nume contin sirul --'Wheel', precum si salariul mediu din acestea si localitatea in care se afla. Coloanele se vor denumi --sugestiv

--Coloanele se vor denumi sugestiv

Pentru această cerere SQL am folosit:

- subcereri sincronizate/corelate pe 3 tabele
- funcția AVG pentru a calcula media aritmetică a salariilor

```
SELECT id_service, nume_service,
(SELECT round(AVG(salariu),2)
FROM angajat
WHERE s.id_service = id_service) "Salariu Mediu",
(SELECT localitate
FROM locatie
WHERE id_locatie = s.id_locatie) "Localitate"
FROM services s
WHERE nume_service LIKE '%Wheel%';
```



Query Result x

SQL | All Rows Fetched: 2 in 0.004 seconds

	ID_SERVICE	NUME_SERVICE	Salariu Mediu	Localitate
1	102	Crazy-Wheel	11183.33	Constanta
2	104	WheelUp	18116.67	Bucuresti

--4. Scrieti o cerere SQL care numara job-urile diferite din fiecare service si afiseaza numarul acestora doar daca este mai mare de 2, alaturi de codul service-ului si de numele acestuia.

Pentru această cerere SQL am folosit:

- funcția COUNT
- gruparea datelor prin GROUP BY
- HAVING ca funcție pe grup

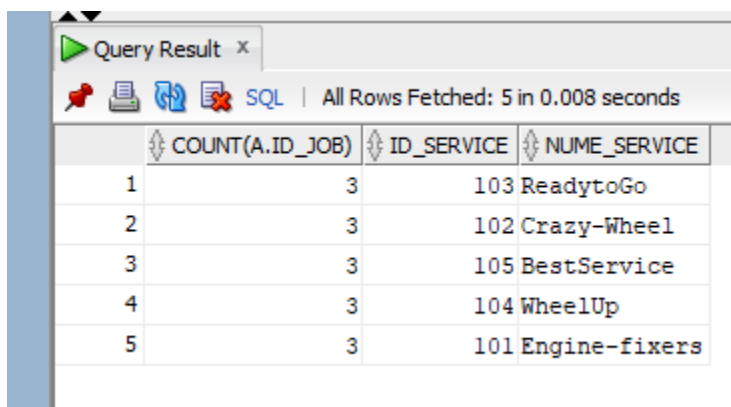
```
SELECT COUNT(a.id_job), s.id_service, s.nume_service
```

```
FROM jobs j JOIN angajat a ON (a.id_job = j.id_job)
```

```
JOIN services s ON(a.id_service = s.id_service)
```

```
GROUP BY s.id_service, s.nume_service
```

```
HAVING COUNT(a.id_job) > 2;
```



	COUNT(A.ID_JOB)	ID_SERVICE	NUME_SERVICE
1	3	103	ReadytoGo
2	3	102	Crazy-Wheel
3	3	105	BestService
4	3	104	WheelUp
5	3	101	Engine-fixers

--5. Scrieti o cerere SQL utilizand clauza WITH pentru a obtine suma salariilor din fiecare service in functie de job_id

Pentru aceasta cerere SQL am folosit:

- Clauza WITH
- Ordonarea datelor ORDER BY

```
WITH valoare_job AS (SELECT nume_job, SUM(salariu) AS total
```

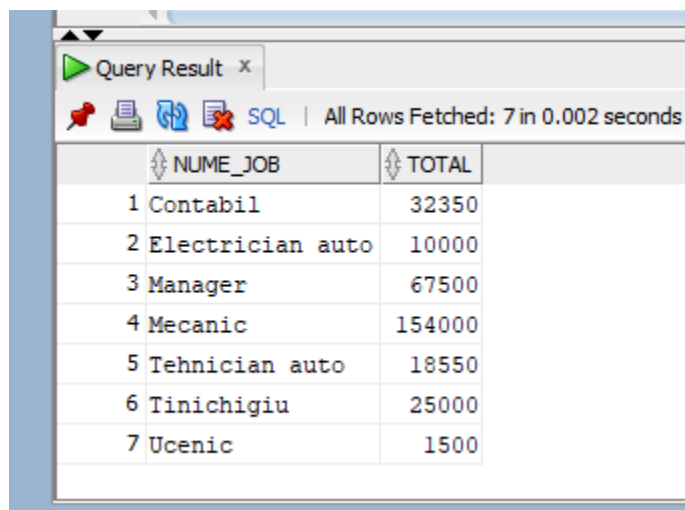
```
FROM jobs j join angajat a ON(j.id_job = a.id_job)
```

```
GROUP BY nume_job)
```

```
SELECT *
```

FROM valoare_job

order by nume_job;



	NUME_JOB	TOTAL
1	Contabil	32350
2	Electrician auto	10000
3	Manager	67500
4	Mecanic	154000
5	Tehnician auto	18550
6	Tinichigiu	25000
7	Ucenic	1500

--6. Scrieti o cerere SQL care sa afiseze numele, prenumele, data_angajarii si salariul initial, alaturi de salariul marit cu

--50%(pentru angajatii care s-au angajat in anii 2000 sau 2001)

--si salariul micorat cu 10% pentru angajatii care s-au angajat in anii 2002, 2003 sau 2004.

--Coloanele se vor denumi sugestiv, iar in cazul in care nu s-au produs modificari ale

--salariului in vreuna dintre noile coloane, se va trece salariul initial.

Pentru această cerere SQL am folosit:

- DECODE
- NVL
- CASE

SELECT nume_angajat, prenume_angajat, data_angajarii, salariu,

NVL((DECODE(TO_CHAR(data_angajarii, 'yyyy'), '2000', salariu * 1.5, '2001', salariu* 1.5)),
salariu) "Salariu marit",

CASE TO_CHAR(data_angajarii, 'yyyy')

WHEN '2002' THEN salariu * 0.9

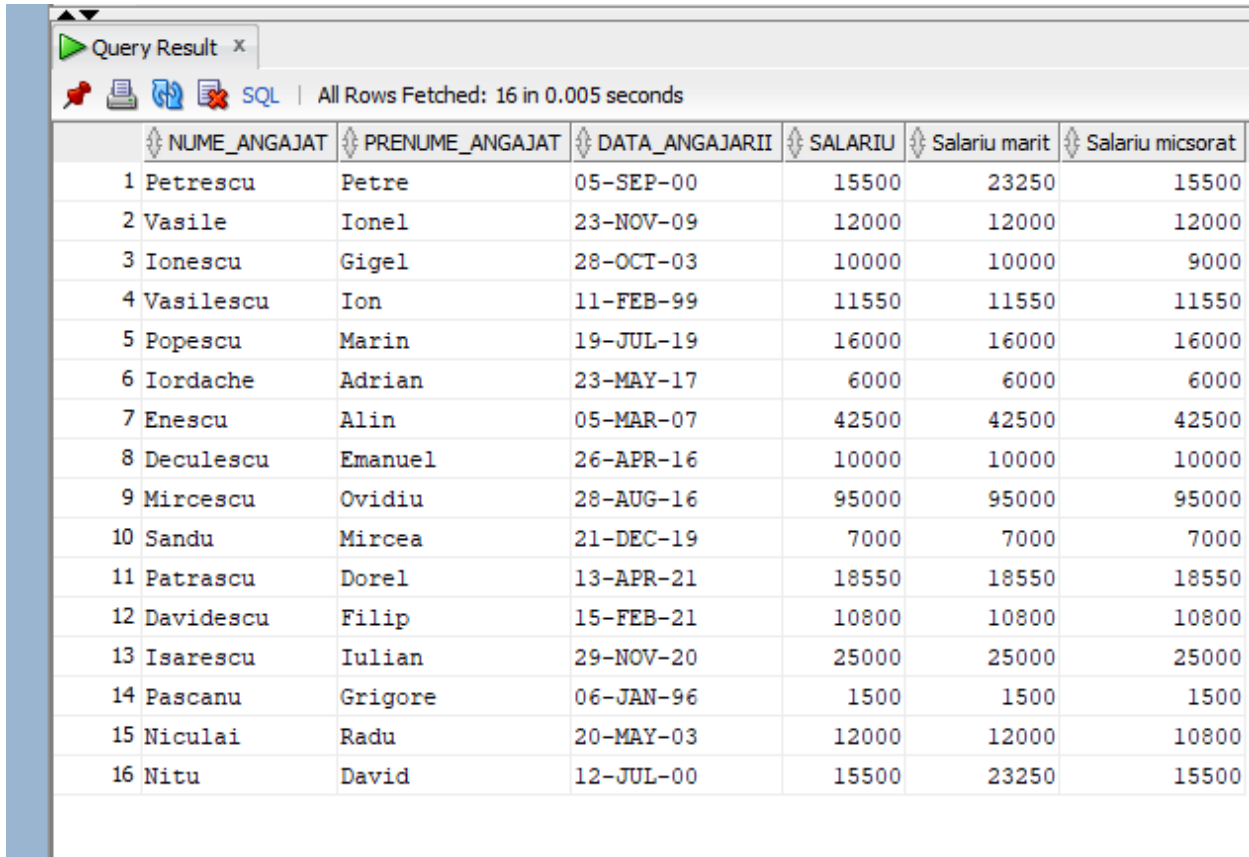
WHEN '2003' THEN salariu * 0.9

WHEN '2004' THEN salariu * 0.9

ELSE salariu

END "Salariu micsorat"

FROM angajat;



	NUME_ANGAJAT	PRENUME_ANGAJAT	DATA_ANGAJARII	SALARIU	Salariu marit	Salariu micsorat
1	Petrescu	Petre	05-SEP-00	15500	23250	15500
2	Vasile	Ionel	23-NOV-09	12000	12000	12000
3	Ionescu	Gigel	28-OCT-03	10000	10000	9000
4	Vasilescu	Ion	11-FEB-99	11550	11550	11550
5	Popescu	Marin	19-JUL-19	16000	16000	16000
6	Iordache	Adrian	23-MAY-17	6000	6000	6000
7	Enescu	Alin	05-MAR-07	42500	42500	42500
8	Deculescu	Emanuel	26-APR-16	10000	10000	10000
9	Mircescu	Ovidiu	28-AUG-16	95000	95000	95000
10	Sandu	Mircea	21-DEC-19	7000	7000	7000
11	Patrascu	Dorel	13-APR-21	18550	18550	18550
12	Davidescu	Filip	15-FEB-21	10800	10800	10800
13	Isarescu	Iulian	29-NOV-20	25000	25000	25000
14	Pascanu	Grigore	06-JAN-96	1500	1500	1500
15	Niculai	Radu	20-MAY-03	12000	12000	10800
16	Nitu	David	12-JUL-00	15500	23250	15500

ACTUALIZARI/SUPRIMARI DE DATE UTILIZAND SUBCERERI

- Actualizarea atributului valoare (la media aritmetică a tuturor valorilor) din tabelul FURNIZOR dacă este mai mare decât media valorilor.

UPDATE furnizor

SET (valoare) = (SELECT AVG(valoare)

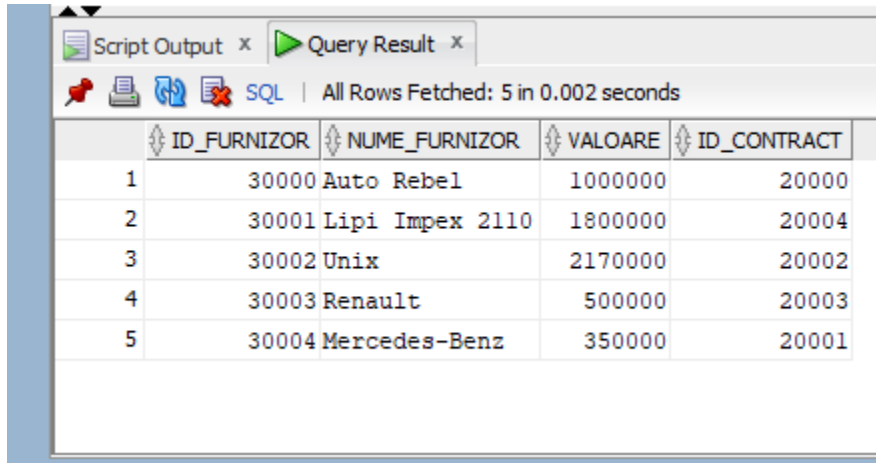
FROM FURNIZOR)

WHERE valoare > (SELECT AVG(valoare)

FROM FURNIZOR);


```
SELECT* FROM FURNIZOR;
```

```
ROLLBACK;
```



The screenshot shows a SQL query result window with the title 'Query Result'. It displays the results of a query on the FURNIZOR table. The window has a toolbar with icons for saving, refreshing, and other database operations. The status bar indicates 'All Rows Fetched: 5 in 0.002 seconds'. The table has five columns: ID_FURNIZOR, NUME_FURNIZOR, VALOARE, and ID_CONTRACT. The data is as follows:

ID_FURNIZOR	NUME_FURNIZOR	VALOARE	ID_CONTRACT
1	30000 Auto Rebel	1000000	20000
2	30001 Lipi Impex 2110	1800000	20004
3	30002 Unix	2170000	20002
4	30003 Renault	500000	20003
5	30004 Mercedes-Benz	350000	20001

2. Actualizarea atributului durata (la media aritmetică a acestora) din tabelul CONTRACT dacă este mai mare decât media aritmetică a tuturor duratelor.

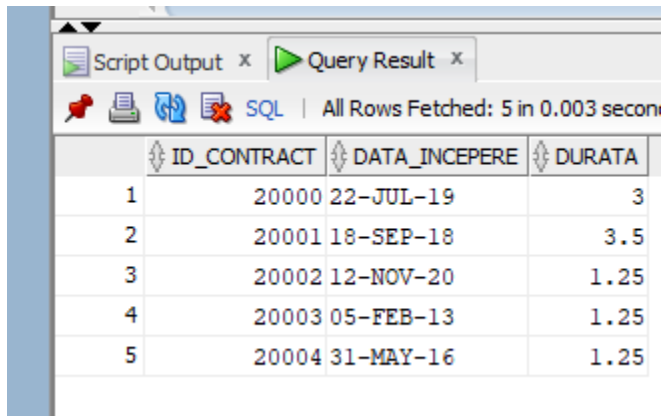
```
UPDATE contract
```

```
SET (durata) = (SELECT MIN(durata)  
FROM contract)
```

```
WHERE durata > (SELECT AVG(durata)  
FROM contract);
```

```
SELECT* FROM CONTRACT;
```

```
ROLLBACK;
```



The screenshot shows a SQL query result window with the title 'Query Result'. It displays the results of a query on the CONTRACT table. The window has a toolbar with icons for saving, refreshing, and other database operations. The status bar indicates 'All Rows Fetched: 5 in 0.003 seconds'. The table has three columns: ID_CONTRACT, DATA_INCEPERE, and DURATA. The data is as follows:

ID_CONTRACT	DATA_INCEPERE	DURATA
1	20000 22-JUL-19	3
2	20001 18-SEP-18	3.5
3	20002 12-NOV-20	1.25
4	20003 05-FEB-13	1.25
5	20004 31-MAY-16	1.25

3. Suprimarea datelor din tabelul monteaza dacă salariul angajatului este mai mare decât 15000 și este angajat după data de 1 ianuarie 2010.

--am facut 10 inserari in acest tabel, deci delete-ul a sters 2 dintre ele.

```
DELETE FROM monteaza
```

```
WHERE id_angajat IN
```

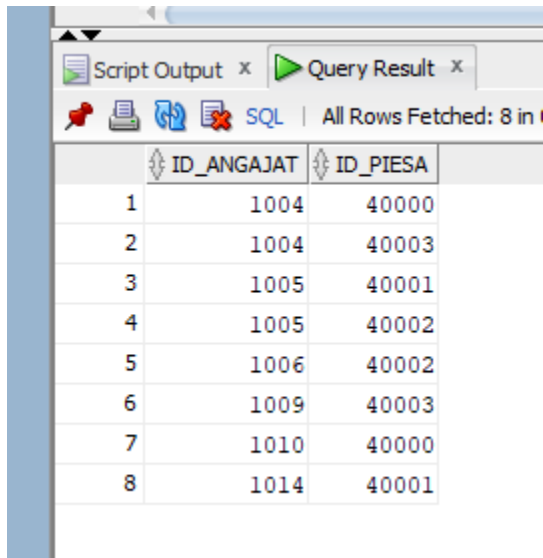
```
(SELECT a.id_angajat
```

```
FROM angajat a
```

```
WHERE salariu > 15000 and data_angajarii > '01-JAN-2010');
```

```
SELECT* FROM MONTEAZA;
```

```
ROLLBACK;
```



The screenshot shows a SQL query result window with two tabs: 'Script Output' and 'Query Result'. The 'Query Result' tab is active, displaying a table with 8 rows and 3 columns: 'ID_ANGAJAT', 'ID_PIESA', and an unlabeled column. The data is as follows:

	ID_ANGAJAT	ID_PIESA	
1	1004	40000	
2	1004	40003	
3	1005	40001	
4	1005	40002	
5	1006	40002	
6	1009	40003	
7	1010	40000	
8	1014	40001	

ALGEBRA RELATIONALA

Scrieti o cerere SQL care afiseaza codul, numele, prenumele, numele job-ului angajatilor care lucreaza ca mecanici in Bucuresti. Sa se afiseze si anul infiintarii service-ului in care lucreaza (0 daca este null), numele acestuia si localitate in care se afla.

```
SELECT id_angajat, nume_angajat, prenume_angajat, NVL(an_infiintare, 0), nume_service,  
localitate, nume_job
```

```
FROM angajat a JOIN services s ON(a.id_service = s.id_service)
```

```
JOIN locatie l ON(s.id_locatie = l.id_locatie)
```

```
JOIN jobs j ON(a.id_job = j.id_job)
```

```
WHERE nume_job = 'Mecanic' AND LOWER(localitate) = 'bucuresti';
```

Expresia algebrica:

R1 = PROJECT(ANGAJAT, id_angajat, nume_angajat, prenume_angajat, id_service, id_job)

R2 = SEMIJOIN(R1, SERVICES, id_service)

R3 = SELECT(LOCATIE, LOWER(localitate) = 'bucuresti')

R4 = PROJECT(R3, id_locatie, localitate)

R5 = SEMIJOIN(R2, R4, id_locatie)

R6 = SELECT(JOBS, nume_job = 'Mecanic')

R7 = PROJECT(R6, id_job, nume_job)

R8 = SEMIJOIN(R5, R7, id_job)

REZULTAT = R9 = PROJECT(R8, id_angajat, nume_angajat, prenume_angajat,
NVL(an_infiintare, 0), nume_service, localitate, nume_job)

Cererea este deja optima deoarece:

- SELECT-urile se fac cat mai devreme, astfel se asigura ca se lucreaza doar cu intrarile necesare (cele care respecta conditiile cerute)
- PROJECT-urile se fac cat mai devreme, astfel se elimina attributele nefolositoare din fiecare tabel cu care se lucreaza
- SEMIJOIN-urile se fac pe datele care conteaza, fiind astfel mult mai putine decat initial

Arbore algebric:

