Hanghicel Razvan-Mihai

Grupa 141

Proiect Baze de Date

1. Descrierea modelului real, a utilității acestuia și a regulilor de funcționare.

Modelul acestei baze de date va gestiona un lant de restaurante. Fiecare restaurant are contract cu un furnizor care se ocupă de aprovizionarea cu materii prime. În fiecare restaurant putem regasi 2 tipuri de angajați: bucatari si ospatari.

Se știe ca fiecare restaurant are doar o singura locatie, deci restaurantele vor fi unice atat prin nume, cat si prin locatie. În fiecare restaurant o sa găsim 2 tipuri de meniuri: meniu pentru mancare si meniu pentru băuturi, meniul pentru fiecare restaurant fiind unic.

Un client poate fi servit de mai mulți ospatari și un ospatar poate servi mai multi clienti

Un client poate avea mai multe comenzi, fiecare comanda fiind alcatuita din mai multe subcomenzi, în care o subcomanda are următoarea formă: cantitate produs din meniu + produs: 2 aripioare de pui, 3 sticle de vin. După ce un client este satisfăcut și dorește sa paraseasca restaurantul, se va genera nota de plata pentru consumația sa.

1. Prezentarea constrângerilor (restrictii, reguli) impuse asupra modelului.

Un ospatar poate sa servească doar produsele care se regasesc în interiorul restaurantului în care lucrează.

Un client poate comanda doar produsele care se afla în meniul restaurantului în care se afla.

Modelul de date respecta anumite restricții de funcționare:

* Un restaurant are o locatie, iar o locatie are un restaurant
* Un meniu apartine unui restaurant, iar un restaurant are mai multe meniuri
* Un meniu are mai multe produse, iar produsele se regăsesc într-un singur meniu
* Un angajat poate lucra doar într-un singur restaurant
* Un ospatar poate lua mai multe comenzi
* Un client poate fi servit de mai mulți ospatari, iar un ospatar poate servi mai mulți clienți
* Un client poate cere mai multe comenzi, iar o comanda poate fi cerută de mai mulți clienți
* O comanda are o singura factura, iar o factura aparține unei comenzi
* Un furnizor poate colabora cu mai multe restaurante, iar un restaurant poate colabora cu un singur furnizor

1. Descrierea entitatilor, incluzand precizarea cheii primare.

Entitățile pentru aceasta baza de date sunt:

* RESTAURANT
* LOCATIE
* FURNIZOR
* COMANDA
* SUBCOMANDA
* MENIU
* PRODUS
* ANGAJAT
* BUCATAR
* OSPATAR
* NOTA\_PLATA
* CLIENT

Descriere entități + precizarea cheii primare

* RESTAURANT - locul unde clientul vine, comanda, mananca, plateste si pleaca. Cheia primara pentru aceasta entitate: **#id\_restaurant**
* LOCAȚIE - locul unde este amplasat restaurantul. Cheia primara pentru aceasta entitate: **#id\_locatie**
* FURNIZOR - organizație care se ocupă cu aprovizionarea cu materii prime pentru restaurante. Cheia primara pentru aceasta entitate: **#id\_furnizor**
* COMANDA - total subcomenzi pentru un client. Cheia primara pentru aceasta entitate: **#id\_comanda**
* SUBCOMANDA - o comanda de forma cantitate dintr-un produs + produs. Cheia primara pentru aceasta entitate: **#id\_subcomanda**
* MENIU - entitate care reține toate felurile de mâncare și tipurile de bauturi dintr-un restaurant. Cheia primara pentru aceasta entitate: **#id\_meniu**
* PRODUS - opțiune pe care un client o poate alege dintr-un meniu. Cheia primara pentru aceasta entitate: **#id\_produs**
* ANGAJAT - persoana care lucrează într-un restaurant. Cheia primara pentru aceasta entitate: **#id\_angajat**
* BUCATAR - subentitate a entitatii ANGAJAT, care are informații legate de bucatari. Cheia primara pentru aceasta subentitate: **#id\_angajat**
* OSPATAR - subentitate a entitatii ANGAJAT, care are informații legate de ospatari. Cheia primara pentru aceasta subentitate: **#id\_angajat**
* NOTA\_PLATA - document pe care sunt trecute produsele consumate, cantitatea din fiecare, prețul per bucata si totalul de plata pe care îl are clientul de achitat. Cheia primara pentru aceasta entitate: **#id\_nota\_plata**
* CLIENT - persoana ce comanda în incinta unui restaurant. Cheia primara pentru aceasta entitate: **#id\_client**

1. Descrierea relațiilor, incluzand precizarea cardinalitatii acestora

Pentru fiecare relație o sa specificam entitățile care intra în construcția ei, cardinalitatea minima si maxima.

FURNIZOR aduce materii prime RESTAURANT - leaga entitatile FURNIZOR si RESTAURANT, reflectand legatura dintre acestea. Cardinalitatea minima este 1:1 (un furnizor trebuie sa aducă materii prime pentru cel puțin un restaurant, iar un restaurant trebuie sa aibă cel puțin un furnizor). Cardinalitatea maxima este 1:M (un furnizor poate să aducă materii prime pentru mai multe restaurante, iar un restaurant poate avea un singur furnizor).

RESTAURANT are LOCATIE - leaga entitatile RESTAURANT si LOCATIE, reflectand legatura dintre acestea. Cardinalitatea minima este 1:1 (un restaurant trebuie să aibă cel puțin o locație, iar o locație trebuie sa aibă cel puțin un restaurant). Cardinalitatea maxima este 1:1 (o locatie poate avea cel mult un restaurant, iar un restaurant poate fi într-o singura locație).

ANGAJAT lucrează într-un RESTAURANT - leaga entitatile ANGAJAT si RESTAURANT, reflectand legatura dintre acestea. Cardinalitatea minima este 1:1 (într-un restaurant trebuie să lucreze cel puțin un angajat, iar un angajat trebuie sa lucreze în cel puțin un restaurant). Cardinalitatea maxima este M:1 (un angajat lucrează în cel mult un restaurant, iar un restaurant poate avea mai mulți angajați).

RESTAURANT are MENIU - leaga entitatile RESTAURANT si MENIU, reflectand legatura dintre acestea. Cardinalitatea minima este 1:1 (un restaurant trebuie să aibă cel puțin un meniu, iar un meniu trebuie sa fie în cel puțin un restaurant). Cardinalitatea maxima este 1:M (un restaurant are mai multe meniuri, iar un meniu trebuie să aparțină cel mult unui restaurant).

MENIU are PRODUS - leaga entitatile MENIU si PRODUS, reflectand legatura dintre acestea. Cardinalitatea minima este 1:1 (un meniu trebuie să aibă cel puțin un produs, iar un produs trebuie sa fie în cel puțin un meniu). Cardinalitatea maxima este 1:M (un meniu are mai multe produse, iar un produs este într-un meniu).

PRODUS are SUBCOMANDA - leaga entitatile PRODUS si SUBCOMANDA, reflectand legatura dintre acestea. Cardinalitatea minima este 1:1 (un produs trebuie să fie în cel puțin o subcomanda, iar o subcomanda trebuie să aibă cel puțin un produs). Cardinalitatea maxima este 1:1 (o subcomanda poate avea cel mult un produs, iar un produs poate fi într-o singura subcomanda).

COMANDA are SUBCOMANDA - leaga entitatile COMANDA si SUBCOMANDA, reflectand legatura dintre acestea. Cardinalitatea minima este 1:1 (o subcomanda trebuie să fie în cel puțin o comanda, iar o comanda trebuie să aibă cel puțin o subcomanda). Cardinalitatea maxima este M:M (o comanda poate avea mai multe subcomenzi, iar o subcomanda poate fi în mai multe comenzi).

COMANDA are NOTA\_PLATA - leaga entitatile COMANDA si NOTA\_PLATA, reflectand legatura dintre acestea. Cardinalitatea minima este 1:1 (o comanda trebuie să aibă cel puțin o nota de plata, iar o nota de plata este data de cel putin o comanda). Cardinalitatea maxima este 1:1 (o nota de plata este data de maxim o comanda, iar o comanda are maxim o nota de plata).

CLIENT are COMANDA - leaga entitatile CLIENT si COMANDA, reflectand legatura dintre acestea. Cardinalitatea minima este 1:1 (un client trebuie să ceara cel puțin o comanda, iar o comanda trebuie sa fie cerută de cel puțin un client). Cardinalitatea maxima este M:M (un client poate cere mai multe comenzi, iar o comanda poate fi cerută de mai mulți clienți).

COMANDA are OSPATAR - leaga entitatile COMANDA si OSPATAR, reflectand legatura dintre acestea. Cardinalitatea minima este 1:1 (o comanda trebuie să fie preluată cel puțin un ospatar, iar un ospatar trebuie sa preia cel puțin o comanda). Cardinalitatea maxima este M:M (o comanda poate fi preluată mai mulți ospatari, iar un ospatar poate prelua mai multe comenzi).

OSPATAR are CLIENT - leaga entitatile OSPATAR si CLIENT, reflectand legatura dintre acestea. Cardinalitatea minima este 1:1 (un ospatar trebuie să servească cel puțin un client, iar un client trebuie sa fie servit de cel puțin un ospatar). Cardinalitatea maxima este M:M (un client poate fi servit de mai mulți ospatari, iar un ospatar poate servi mai mulți clienți).

Pentru a rezolva relațiile de tipul many-to-many si relatiile de tip 3, o sa ne folosim de tabele asociative, care vor fi atribuite următoarelor tabele:

* COMANDA, CLIENT, OSPATAR avem relație many-to-many intre toate trei, deci va fi o relație de tip 3. Pentru asta vom crea un tabel asociativ COMANDA\_CLIENT\_OSPATAR, care va avea ca cheie primara compunerea cheilor primare din cele 3 tabele
* COMANDA si SUBCOMANDA avem relatie many-to-many. Pentru asta vom crea un tabel asociativ COMANDA\_SUBCOMANDA, care va avea ca cheie primara compunerea cheilor primare din cele 2 tabele

1. Descrierea atributelor, incluzand tipul de date și eventualele constrangeri, valori implicite, valori posibile ale atributelor

Structura entitatilor:

* RESTAURANT:
  + **#id\_restaurant**: number(4), primary key, id-ul unic al restaurantului
  + **Id\_locatie**: number(4), foreign key
  + **Id\_furnizor**: number(4), foreign key
  + **nume**: varchar(25) not null, numele restaurantului
* LOCATIE:
  + **#id\_locatie**: number(4), primary key, id-ul unic al locatiei
  + **adresa**: varchar(25) not null, locul unde se afla restaurantul
* FURNIZOR:
  + **#id\_furnizor**: number(4), primary key, id-ul unic al furnizorului
  + **nume**: varchar(25) not null, numele furnizorului
* COMANDA:
  + **#id\_comanda**: number(4), primary key, id-ul unic al comenzii
* SUBCOMANDA:
  + **#id\_subcomanda**: number(4), primary key, id-ul unic al subcomenzii
  + **id\_produs**: number(4), foreign key
  + **cantitate**: number(2) not null, cantitatea din acel produs care a fost comandata
* MENIU:
  + **#id\_meniu**: number(4), primary key, id-ul unic al meniului
  + **id\_restaurant**: number(4), foreign key
  + **nume**: varchar(25) not null, numele meniului
* PRODUS:
  + **#id\_produs**: number(4), primary key, id-ul unic al produsului
  + **Id\_meniu**: varchar(4), foreign key
  + **nume**: varchar(25) not null, numele produsului
* ANGAJAT:
  + **#id\_angajat**: number(4), primary key, id-ul unic al angajatului
  + **id\_restaurant**: number(4), foreign key
  + **nume**: varchar(25) not null, numele angajatului
  + **prenume**: varchar(25) not null, prenumele angajatului
  + **salariu**: number(6,2) not null, salariul angajatului
  + **data\_angajarii**: date, data la care a fost angajat
  + **tip**: varchar(25) not null, poate fi “bucatar” sau “ospatar”
* NOTA\_PLATA:
  + **#id\_nota\_plata**: number(4), primary key, id-ul unic al platii
  + **Id\_comanda**: number(4), foreign key
* CLIENT:
  + **#id\_client**: number(4), primary key, id-ul unic al clientului
  + **nume**: varchar(25) not null, numele clientului
  + **prenume**: varchar(25) not null, prenumele clientului

Structura tabelelor asociative:

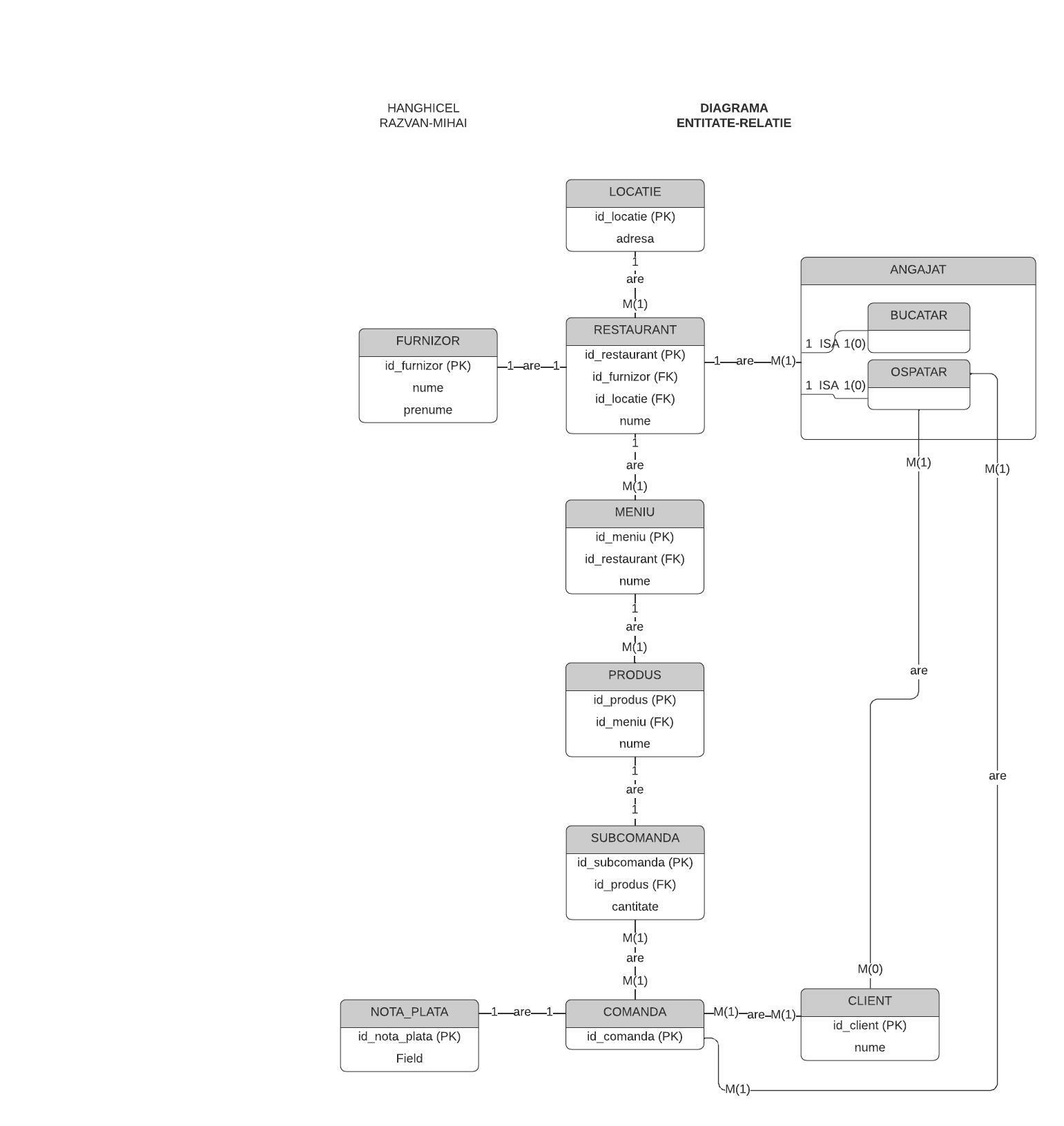
* COMANDA\_CLIENT\_OSPATAR:
  + **id\_comanda**: number(4), foreign key
  + **id\_client**: number(4), foreign key
  + **id\_andajat**: number(4), foreign key

Toate aceste chei formează o cheie primara compusă pentru tabelul asociativ COMANDA\_CLIENT\_OSPATAR

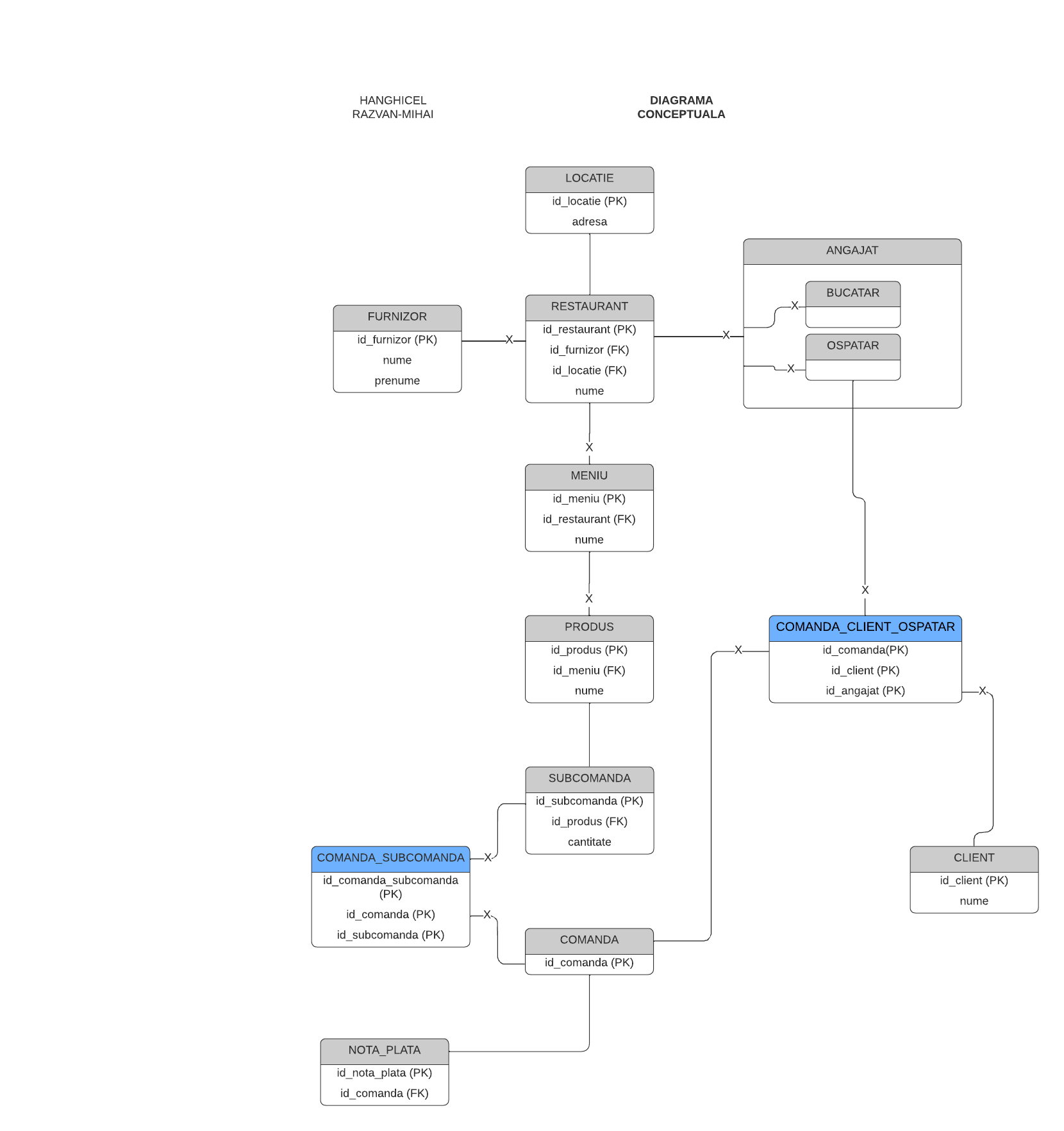
* COMANDA\_SUBCOMANDA:
  + **#id\_comanda\_subcomanda**: number(4), primary key
  + **id\_comanda**: number(4), foreign key
  + **id\_subcomanda**: number(4), foreign key

Toate aceste chei formează o cheie primara compusă pentru tabelul asociativ COMANDA\_SUBCOMANDA

1. Realizarea diagramei entitate-relatie corespunzătoare descrierii de la punctele 3-5



1. Realizarea diagramei conceptuale corespunzătoare diagramei entitate-relatie proiectate la punctul 6. Diagrama conceptuala obtinuta sa contina minimum 6 tabele (fără considerarea subentităților), dintre care cel putin un table asociativ.



1. Enumerarea schemelor relaționale corespunzătoare diagramei conceptuale proiectate la punctul 7

RESTAURANT (#id\_restaurant, id\_furnizor, id\_locatie, nume)

FURNIZOR (#id\_furnizor, nume)

LOCATIE (#id\_locatie, adresa)

MENIU (#id\_meniu, id\_restaurant, nume)

PRODUS (#id\_produs, id\_meniu, nume)

SUBCOMANDA (#id\_subcomanda, id\_produs, cantitate)

COMANDA (#id\_comanda)

COMANDA\_SUBCOMANDA (#id\_comanda\_subcomanda, #id\_comanda, #id\_subcomanda)

NOTA\_PLATA (#id\_nota\_plata, id\_comanda)

ANGAJAT (#id\_angajat, id\_restaurant, nume, prenume, salariu, data\_angajarii, tip)

COMANDA\_CLIENT\_OSPATAR (#id\_comanda, #id\_client, #id\_angajat)

1. Realizarea normalizarii pana la forma normala 3 (FN1-FN3)

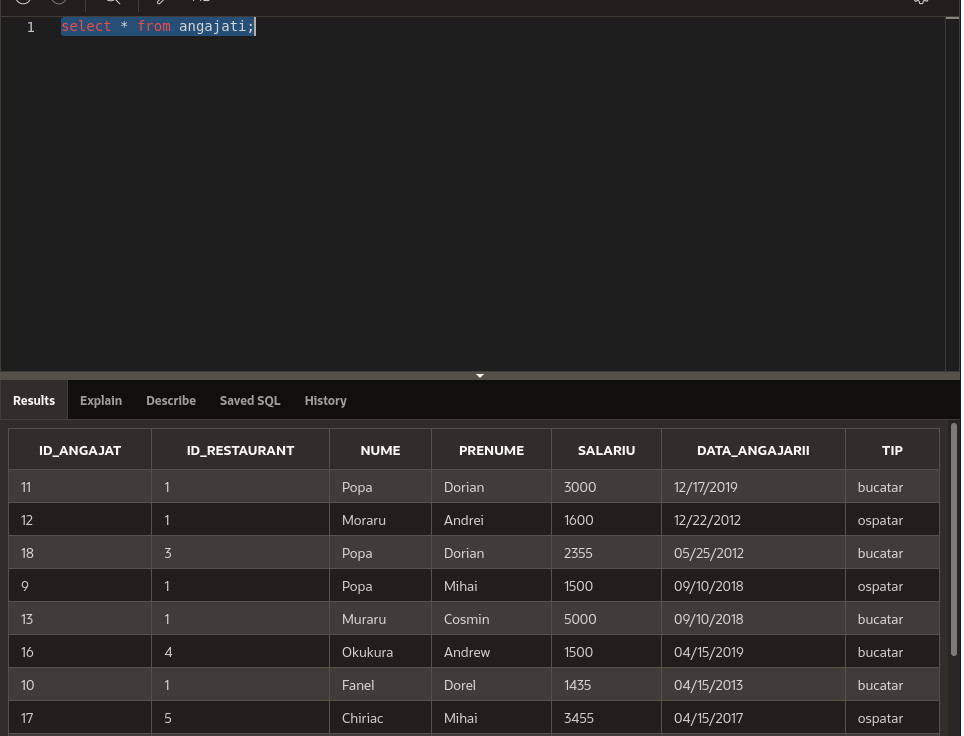
**NORMALIZARE FN1**

O relație este in FN1 dacă domeniile pe care sunt definite atributele relației sunt constituite numai din valori atomice. Un tuplu nu trebuie sa contina atribute sau grupuri de atribute repetitive.

| Angajat |
| --- |
| Popa Dorian 3000 12/17/2019 bucatar 1 |
| Moraru Andrei 1600 12/22/2012 ospatar 1 |
| Popa Dorian 2355 05/25/2012 bucatar 3 |
| Popa Mihai 1500 09/10/2018 ospatar 1 |

Acest tabel făcut de noi nu se afla in FN1

Dar tabelul pe care l-am creat se afla in FN1.



Tabelul se afla in FN1 deoarece fiecare angajat are cheia primara unica, deci nu o sa fie doi angajați cu aceeasi cheie primara.

**NORMALIZARE FN2**

O relație R este in a doua forma normala dacă și numai dacă:

* + relatia R este in FN1
  + fiecare atribut care nu face parte din cheia primara este dependent de intreaga cheie primara

FN2 interzice manifestarea unor dependente functionale parțiale în cadrul relației R

Fie tabelul SUBCOMANDA cu atributele #id\_subcomanda, #id\_comanda, cantitate

| #id\_subcomanda | #id\_comanda | cantitate |
| --- | --- | --- |
| 3 | 5 | 4 |
| 2 | 1 | 1 |
| 3 | 2 | 4 |

Se poate observa că acest tabel nu se afla in FN2, deoarece atributele care nu fac parte din cheie (in acest caz **cantitate**) depind doar de o parte a cheii.

Pentru a trece acest tabel în FN2 o sa folosim regula Casey-Delobel, adică trebuie sa formăm din acest tabel 2 tabele care sa conțină toate datele din tabelul inițial și care au ambele ca cheie primara id\_subcomanda

| #id\_subcomanda | #id\_comanda |
| --- | --- |
| 3 | 5 |
| 2 | 1 |
| 3 | 2 |

| #id\_subcomanda | cantitate |
| --- | --- |
| 3 | 4 |
| 2 | 1 |
| 3 | 4 |

Deci am inlocuit tabelul nostru inițial cu 2 tabele care reprezinta proiectiile tabelului initial, fara a pierde informatii

**NORMALIZARE FN3**

O relatie R este in a treia forma normala daca si numai daca:

* + relatia R este in FN2
  + Fiecare atribut care nu face parte din cheia primara depinde direct de cheia primara

Pentru a putea demonstra FN3, o sa mai adaugam cateva atribute (tip produs, pret)

| #id\_subcomanda | #id\_comanda | tip produs | pret |
| --- | --- | --- | --- |
| 3 | 5 | vinuri | 350 |
| 2 | 1 | sucuri | 250 |
| 3 | 2 | vinuri | 350 |

Putem observa ca tabelul de mai sus este deja in FN2, doar ca nu este in FN3, deoarece pret depinde tranzitiv de cheia primara prin intermediul lui tip produs

Pentru asta o sa utilizăm regula lui Casey Delobel pentru a realiza normalizarea in FN3

| #id\_subcomanda | #id\_comanda | Tip produs |
| --- | --- | --- |
| 3 | 5 | vinuri |
| 2 | 1 | sucuri |
| 3 | 2 | vinuri |

| tip\_mancare | pret |
| --- | --- |
| vinuri | 350 |
| sucuri | 250 |

1. Crearea tabelelor in SQL și inserarea de date coerente în fiecare dintre acestea (minimum 5 înregistrări în fiecare tabel neasociativ; minimum 10 înregistrări în tabelele asociative)
   * Creare si inserare date în tabelul FURNIZOR

CREATE TABLE FURNIZOR

(

id\_furnizor number(4),

nume VARCHAR2(25) NOT NULL

);

ALTER TABLE FURNIZOR

ADD CONSTRAINT pk\_furnizor PRIMARY KEY(id\_furnizor);

CREATE SEQUENCE SEQ\_FURNIZOR

INCREMENT by 1

START WITH 0

MINVALUE 0

MAXVALUE 50

NOCYCLE;

INSERT INTO FURNIZOR(id\_furnizor, nume)

VALUES(SEQ\_FURNIZOR.nextval, 'EMAG');

INSERT INTO FURNIZOR(id\_furnizor, nume)

VALUES(SEQ\_FURNIZOR.nextval, 'FERMA MEA');

INSERT INTO FURNIZOR(id\_furnizor, nume)

VALUES(SEQ\_FURNIZOR.nextval, 'GRADINA BUNICII');

INSERT INTO FURNIZOR(id\_furnizor, nume)

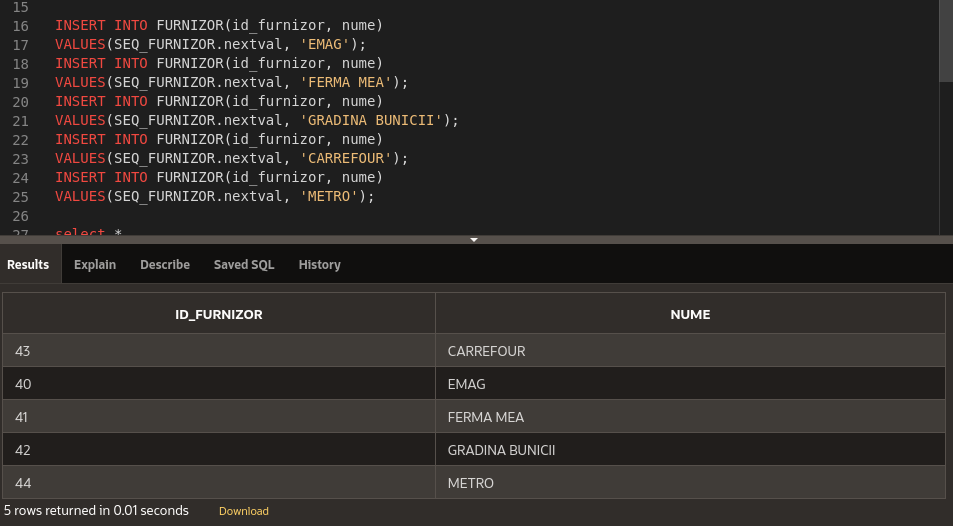
VALUES(SEQ\_FURNIZOR.nextval, 'CARREFOUR');

INSERT INTO FURNIZOR(id\_furnizor, nume)

VALUES(SEQ\_FURNIZOR.nextval, 'METRO');

select \*

from FURNIZOR;



* + Creare si inserare date în tabelul LOCAȚIE

CREATE TABLE LOCATIE

(

id\_locatie number(3) constraint pk\_LOCATIE primary key,

adresa VARCHAR2(60) not null

);

CREATE SEQUENCE SEQ\_LOCATIE

INCREMENT BY 1

START WITH 1

MINVALUE 0

MAXVALUE 100

NOCYCLE

NOCACHE

ORDER;

INSERT INTO LOCATIE(id\_locatie, adresa)

VALUES(SEQ\_LOCATIE.nextval, 'Piata Roman Voda, Roman, 617226'); -- Anturaj

INSERT INTO LOCATIE(id\_locatie, adresa)

VALUES(SEQ\_LOCATIE.nextval, 'Strada Ștefan cel Mare, Roman 617246'); -- Casa Romascana

INSERT INTO LOCATIE(id\_locatie, adresa)

VALUES(SEQ\_LOCATIE.nextval, 'Pietonal Ștefan cel Mare 17, Roman 617135');-- MGH

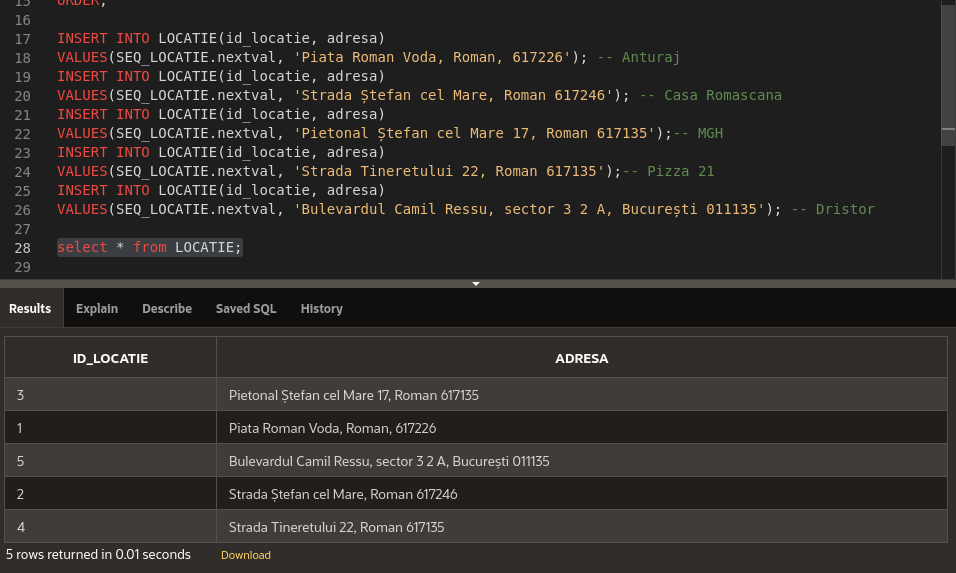
INSERT INTO LOCATIE(id\_locatie, adresa)

VALUES(SEQ\_LOCATIE.nextval, 'Strada Tineretului 22, Roman 617135');-- Pizza 21

INSERT INTO LOCATIE(id\_locatie, adresa)

VALUES(SEQ\_LOCATIE.nextval, 'Bulevardul Camil Ressu, sector 3 2 A, București 011135'); -- Dristor

select \* from LOCATIE;



* + Creare si inserare date în tabelul RESTAURANT

CREATE TABLE RESTAURANT

(

id\_restaurant number(4) constraint pk\_restaurant primary key,

id\_locatie number(4) constraint fk\_loc foreign key(id\_locatie) references LOCATIE(id\_locatie),

id\_furnizor number(4) constraint fk\_colab foreign key(id\_furnizor) references furnizor(id\_furnizor)

nume varchar2(25) not null,

);

CREATE SEQUENCE SEQ\_RESTAURANT

INCREMENT BY 1

START WITH 0

MINVALUE 0

MAXVALUE 100

NOCYCLE

NOCACHE

ORDER;

INSERT INTO RESTAURANT(id\_restaurant, id\_locatie, id\_furnizor, nume)

VALUES(SEQ\_RESTAURANT.nextval, 1, 5, 'Anturaj');

INSERT INTO RESTAURANT(id\_restaurant, id\_locatie, id\_furnizor, nume)

VALUES(SEQ\_RESTAURANT.nextval, 2, 3, 'Casa Romascana');

INSERT INTO RESTAURANT(id\_restaurant, id\_locatie, id\_furnizor, nume)

VALUES(SEQ\_RESTAURANT.nextval, 3, 3, 'MGH');

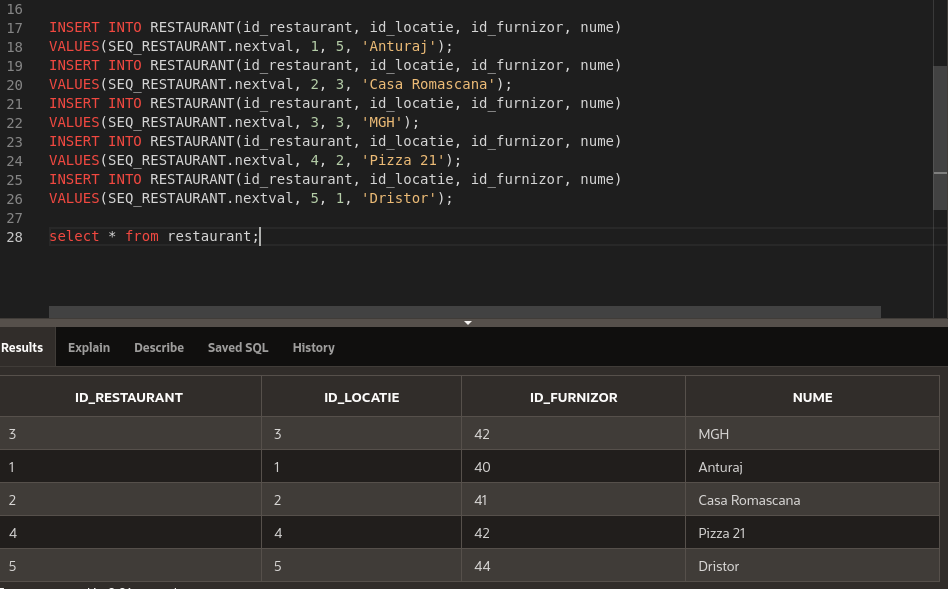
INSERT INTO RESTAURANT(id\_restaurant, id\_locatie, id\_furnizor, nume)

VALUES(SEQ\_RESTAURANT.nextval, 4, 2, 'Pizza 21');

INSERT INTO RESTAURANT(id\_restaurant, id\_locatie, id\_furnizor, nume)

VALUES(SEQ\_RESTAURANT.nextval, 5, 1, 'Dristor');

select \* from restaurant;



* + Creare si inserare date în tabelul CLIENTI

CREATE TABLE CLIENTI

(

id\_client number(4) constraint pk\_clienti primary key,

nume varchar(25) not null,

prenume varchar(25) not null

);

CREATE SEQUENCE SEQ\_CLIENTI

INCREMENT BY 1

START WITH 0

MINVALUE 0

MAXVALUE 100

NOCYCLE

NOCACHE

ORDER;

INSERT INTO CLIENTI(id\_client, nume, prenume)

Values(SEQ\_CLIENTI.nextval, 'Hanghicel', 'Razvan-Mihai');

INSERT INTO CLIENTI(id\_client, nume, prenume)

Values(SEQ\_CLIENTI.nextval, 'Musk', 'Elon');

INSERT INTO CLIENTI(id\_client, nume, prenume)

Values(SEQ\_CLIENTI.nextval, 'Gaitan', 'Rares');

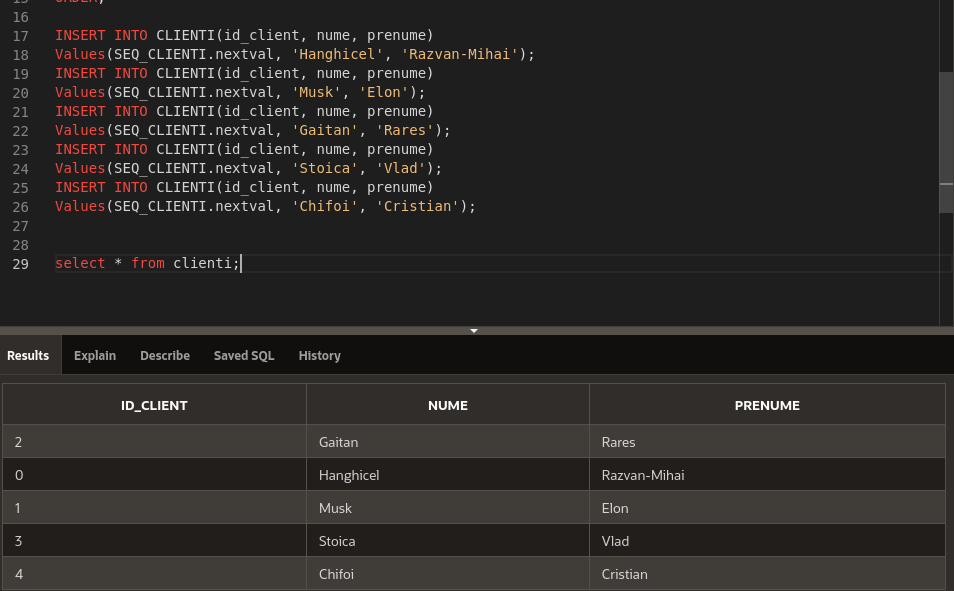
INSERT INTO CLIENTI(id\_client, nume, prenume)

Values(SEQ\_CLIENTI.nextval, 'Stoica', 'Vlad');

INSERT INTO CLIENTI(id\_client, nume, prenume)

Values(SEQ\_CLIENTI.nextval, 'Chifoi', 'Cristian');

select \* from clienti;



* + Creare si inserare date în tabelul ANGAJATI

CREATE TABLE ANGAJATI

(

id\_angajat number(3,0) constraint pk\_ang primary key,

id\_restaurant number(4), constraint fk\_rest FOREIGN key(id\_restaurant) references RESTAURANT(id\_restaurant),

nume varchar2(25) not null,

prenume varchar2(25) not null,

salariu number(5) not null,

data\_angajarii date,

tip varchar(25) not null

);

CREATE SEQUENCE SEQ\_ANGAJATI

INCREMENT BY 1

START WITH 0

MINVALUE 0

MAXVALUE 100

NOCYCLE

NOCACHE

ORDER;

INSERT INTO ANGAJATI(id\_angajat, id\_restaurant, nume, prenume, salariu, tip)

VALUES(SEQ\_ANGAJATI.nextval, 1, 'Popa', 'Mihai', 1500, 'ospatar');

INSERT INTO ANGAJATI(id\_angajat, id\_restaurant, nume, prenume, salariu, tip, data\_angajarii)

VALUES(SEQ\_ANGAJATI.nextval, 1, 'Fanel', 'Dorel', 1435, 'bucatar', to\_date('15-04-2013','dd-mm-yyyy'));

INSERT INTO ANGAJATI(id\_angajat, id\_restaurant, nume, prenume, salariu, tip, data\_angajarii)

VALUES(SEQ\_ANGAJATI.nextval, 1, 'Popa', 'Dorian', 3000, 'bucatar', to\_date('17-12-2019','dd-mm-yyyy'));

INSERT INTO ANGAJATI(id\_angajat, id\_restaurant, nume, prenume, salariu, tip, data\_angajarii)

VALUES(SEQ\_ANGAJATI.nextval, 1, 'Moraru', 'Andrei', 1600, 'ospatar', to\_date('22-12-2012','dd-mm-yyyy'));

INSERT INTO ANGAJATI(id\_angajat, id\_restaurant, nume, prenume, salariu, tip)

VALUES(SEQ\_ANGAJATI.nextval, 1, 'Muraru', 'Cosmin', 5000, 'bucatar');

INSERT INTO ANGAJATI(id\_angajat, id\_restaurant, nume, prenume, salariu, tip, data\_angajarii)

VALUES(SEQ\_ANGAJATI.nextval, 2, 'Danila', 'Prepeleac', 5555, 'ospatar', to\_date('08-01-2021','dd-mm-yyyy'));

INSERT INTO ANGAJATI(id\_angajat, id\_restaurant, nume, prenume, salariu, tip, data\_angajarii)

VALUES(SEQ\_ANGAJATI.nextval, 3, 'Hociung', 'Vlad', 1400, 'ospatar', to\_date('06-12-2007','dd-mm-yyyy'));

INSERT INTO ANGAJATI(id\_angajat, id\_restaurant, nume, prenume, salariu, tip, data\_angajarii)

VALUES(SEQ\_ANGAJATI.nextval, 4, 'Okukura', 'Andrew', 1500, 'bucatar', to\_date('15-04-2019','dd-mm-yyyy'));

INSERT INTO ANGAJATI(id\_angajat, id\_restaurant, nume, prenume, salariu, tip, data\_angajarii)

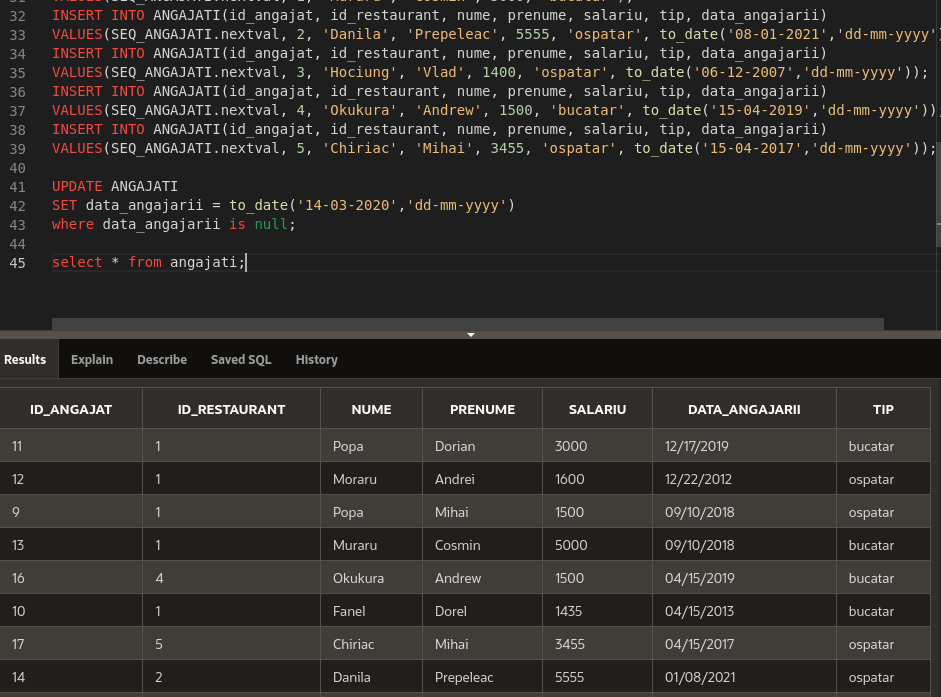
VALUES(SEQ\_ANGAJATI.nextval, 5, 'Chiriac', 'Mihai', 3455, 'ospatar', to\_date('15-04-2017','dd-mm-yyyy'));

UPDATE ANGAJATI

SET data\_angajarii = to\_date('14-03-2020','dd-mm-yyyy')

where data\_angajarii is null;

select \* from angajati;



* Creare si inserare date în tabelul MENIU

CREATE TABLE MENIU

(

id\_meniu number(3) constraint pk\_meniu primary key,

id\_restaurant number(3) not null constraint fk\_rest\_men references RESTAURANT(id\_restaurant),

nume VARCHAR2(25) not null

);

CREATE SEQUENCE SEQ\_MENIU

INCREMENT BY 1

START WITH 0

MINVALUE 0

MAXVALUE 100

NOCYCLE

NOCACHE

ORDER;

INSERT INTO MENIU(id\_meniu, nume, id\_restaurant)

VALUES(SEQ\_MENIU.nextval, 'Meniu cu pui', 1);

INSERT INTO MENIU(id\_meniu, nume, id\_restaurant)

VALUES(SEQ\_MENIU.nextval, 'Meniu de pui', 2);

INSERT INTO MENIU(id\_meniu, nume, id\_restaurant)

VALUES(SEQ\_MENIU.nextval, 'Meniu cu porc', 4);

INSERT INTO MENIU(id\_meniu, nume, id\_restaurant)

VALUES(SEQ\_MENIU.nextval, 'Meniu cu vita', 1);

INSERT INTO MENIU(id\_meniu, nume, id\_restaurant)

VALUES(SEQ\_MENIU.nextval, 'Sucuri', 1);

INSERT INTO MENIU(id\_meniu, nume, id\_restaurant)

VALUES(SEQ\_MENIU.nextval, 'Vinuri', 1);

INSERT INTO MENIU(id\_meniu, nume, id\_restaurant)

VALUES(SEQ\_MENIU.nextval, 'Sucuri', 2);

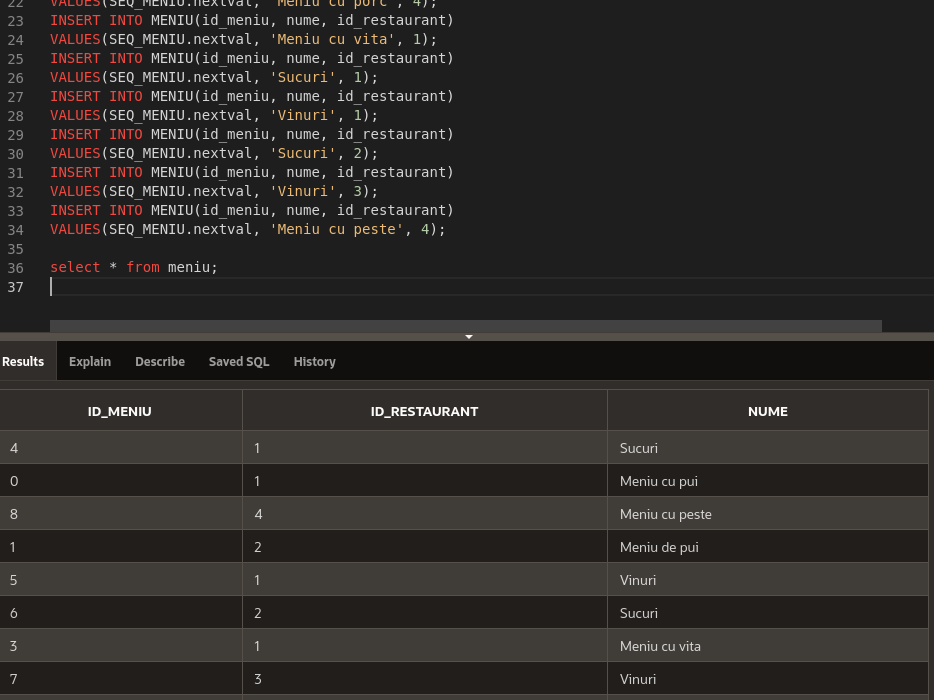
INSERT INTO MENIU(id\_meniu, nume, id\_restaurant)

VALUES(SEQ\_MENIU.nextval, 'Vinuri', 3);

INSERT INTO MENIU(id\_meniu, nume, id\_restaurant)

VALUES(SEQ\_MENIU.nextval, 'Meniu cu peste', 4);

select \* from meniu;



* Creare si inserare date în tabelul PRODUS

CREATE TABLE PRODUS

(

id\_produs number(3) constraint pk\_p\_meniu primary key,

id\_meniu number(3) not null constraint fk\_meniu references MENIU(id\_meniu),

nume VARCHAR2(25) not null

);

CREATE SEQUENCE SEQ\_P\_MENIU

INCREMENT BY 1

START WITH 0

MINVALUE 0

MAXVALUE 100

NOCYCLE

NOCACHE

ORDER;

INSERT INTO PRODUS(id\_produs, nume, id\_meniu)

VALUES(SEQ\_P\_MENIU.nextval, 'Samburesti', 5);

INSERT INTO PRODUS(id\_produs, nume, id\_meniu)

VALUES(SEQ\_P\_MENIU.nextval, 'Fanta', 6);

INSERT INTO PRODUS(id\_produs, nume, id\_meniu)

VALUES(SEQ\_P\_MENIU.nextval, 'Coca Cola', 6);

INSERT INTO PRODUS(id\_produs, nume, id\_meniu)

VALUES(SEQ\_P\_MENIU.nextval, 'Moet', 6);

INSERT INTO PRODUS(id\_produs, nume, id\_meniu)

VALUES(SEQ\_P\_MENIU.nextval, 'Cripsy de pui', 8);

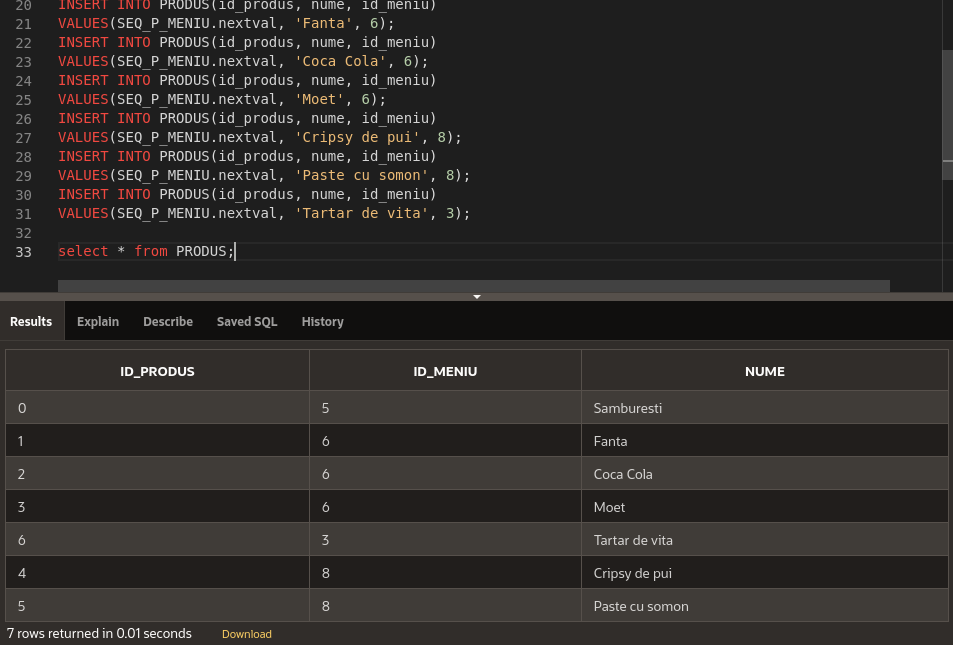
INSERT INTO PRODUS(id\_produs, nume, id\_meniu)

VALUES(SEQ\_P\_MENIU.nextval, 'Paste cu somon', 8);

INSERT INTO PRODUS(id\_produs, nume, id\_meniu)

VALUES(SEQ\_P\_MENIU.nextval, 'Tartar de vita', 3);

select \* from PRODUS;



* Creare si inserare date în tabelul SUBCOMANDA

CREATE TABLE SUBCOMANDA

(

id\_subcomanda number(3) constraint pk\_subcomanda primary key,

id\_produs number(3) not null constraint fk\_produs references PRODUS(id\_produs),

cantitate number(2) not null CHECK (cantitate > 0)

);

CREATE SEQUENCE SEQ\_SUBCOMANDA

INCREMENT BY 1

START WITH 0

MINVALUE 0

MAXVALUE 100

NOCYCLE

NOCACHE

ORDER;

INSERT INTO SUBCOMANDA(id\_subcomanda,cantitate, id\_produs)

Values(SEQ\_CMD\_PROD.nextval, 3, 3);

INSERT INTO SUBCOMANDA(id\_subcomanda,cantitate, id\_produs)

Values(SEQ\_CMD\_PROD.nextval, 1, 1);

INSERT INTO SUBCOMANDA(id\_subcomanda,cantitate, id\_produs)

Values(SEQ\_CMD\_PROD.nextval, 4, 5);

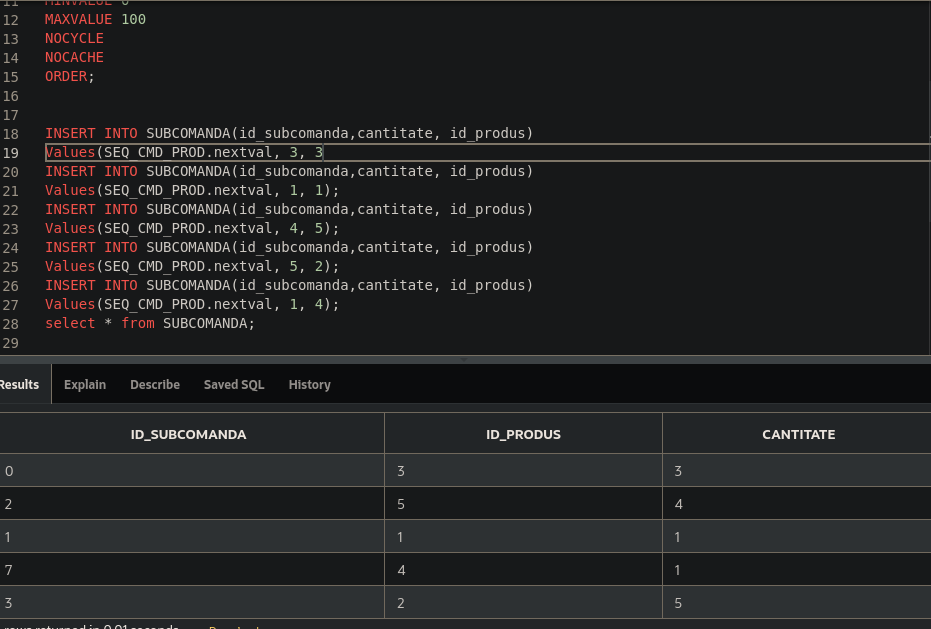
INSERT INTO SUBCOMANDA(id\_subcomanda,cantitate, id\_produs)

Values(SEQ\_CMD\_PROD.nextval, 5, 2);

INSERT INTO SUBCOMANDA(id\_subcomanda,cantitate, id\_produs)

Values(SEQ\_CMD\_PROD.nextval, 1, 4);

select \* from SUBCOMANDA;



* Creare si inserare date în tabelul COMANDA

CREATE TABLE COMANDA

(

id\_comanda number(3) constraint pk\_comanda primary key

);

CREATE SEQUENCE SEQ\_COMANDA

INCREMENT BY 1

START WITH 0

MINVALUE 0

MAXVALUE 100

NOCYCLE

NOCACHE

ORDER;

INSERT INTO COMANDA(id\_comanda)

VALUES(SEQ\_COMANDA.nextval);

INSERT INTO COMANDA(id\_comanda)

VALUES(SEQ\_COMANDA.nextval);

INSERT INTO COMANDA(id\_comanda)

VALUES(SEQ\_COMANDA.nextval);

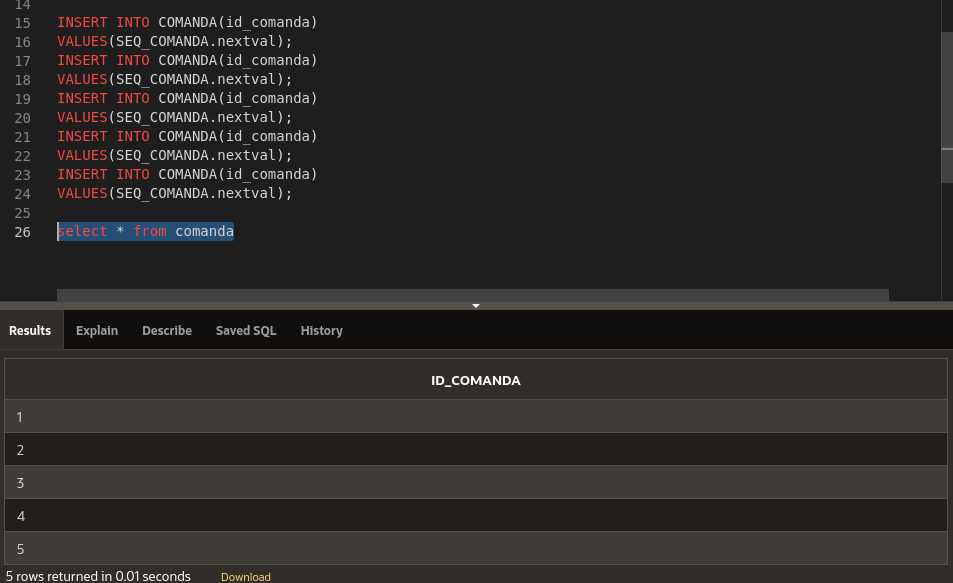
INSERT INTO COMANDA(id\_comanda)

VALUES(SEQ\_COMANDA.nextval);

INSERT INTO COMANDA(id\_comanda)

VALUES(SEQ\_COMANDA.nextval);

select \* from comanda



* Creare si inserare date în tabelul COMANDA\_SUBCOMANDA

CREATE TABLE COMANDA\_SUBCOMANDA

(

id\_comanda\_subcomanda number(3) not null,

id\_comanda number(3) not null constraint fk\_comanda1 references COMANDA(id\_comanda),

id\_subcomanda number(3) not null constraint fk\_subcomanda1 references SUBCOMANDA(id\_subcomanda),

constraint pk\_COMANDA\_SUBCOMANDA primary key (id\_comanda\_subcomanda, id\_comanda, id\_subcomanda)

);

select \* from comanda\_subcomanda

CREATE SEQUENCE SEQ\_COMANDA\_SUBCOMANDA

INCREMENT BY 1

START WITH 0

MINVALUE 0

MAXVALUE 100

NOCYCLE

NOCACHE

ORDER;

select \* from COMANDA\_SUBCOMANDA;

INSERT INTO COMANDA\_SUBCOMANDA(id\_comanda\_subcomanda, id\_comanda, id\_subcomanda)

Values(SEQ\_COMANDA\_SUBCOMANDA.nextval,1,1);

INSERT INTO COMANDA\_SUBCOMANDA(id\_comanda\_subcomanda, id\_comanda, id\_subcomanda)

Values(SEQ\_COMANDA\_SUBCOMANDA.nextval,3,2);

INSERT INTO COMANDA\_SUBCOMANDA(id\_comanda\_subcomanda, id\_comanda, id\_subcomanda)

Values(SEQ\_COMANDA\_SUBCOMANDA.nextval,4,3);

INSERT INTO COMANDA\_SUBCOMANDA(id\_comanda\_subcomanda, id\_comanda, id\_subcomanda)

Values(SEQ\_COMANDA\_SUBCOMANDA.nextval,3,3);

INSERT INTO COMANDA\_SUBCOMANDA(id\_comanda\_subcomanda, id\_comanda, id\_subcomanda)

Values(SEQ\_COMANDA\_SUBCOMANDA.nextval,3,1);

INSERT INTO COMANDA\_SUBCOMANDA(id\_comanda\_subcomanda, id\_comanda, id\_subcomanda)

Values(SEQ\_COMANDA\_SUBCOMANDA.nextval,4,3);

INSERT INTO COMANDA\_SUBCOMANDA(id\_comanda\_subcomanda, id\_comanda, id\_subcomanda)

Values(SEQ\_COMANDA\_SUBCOMANDA.nextval,1,2);

INSERT INTO COMANDA\_SUBCOMANDA(id\_comanda\_subcomanda, id\_comanda, id\_subcomanda)

Values(SEQ\_COMANDA\_SUBCOMANDA.nextval,2,3);

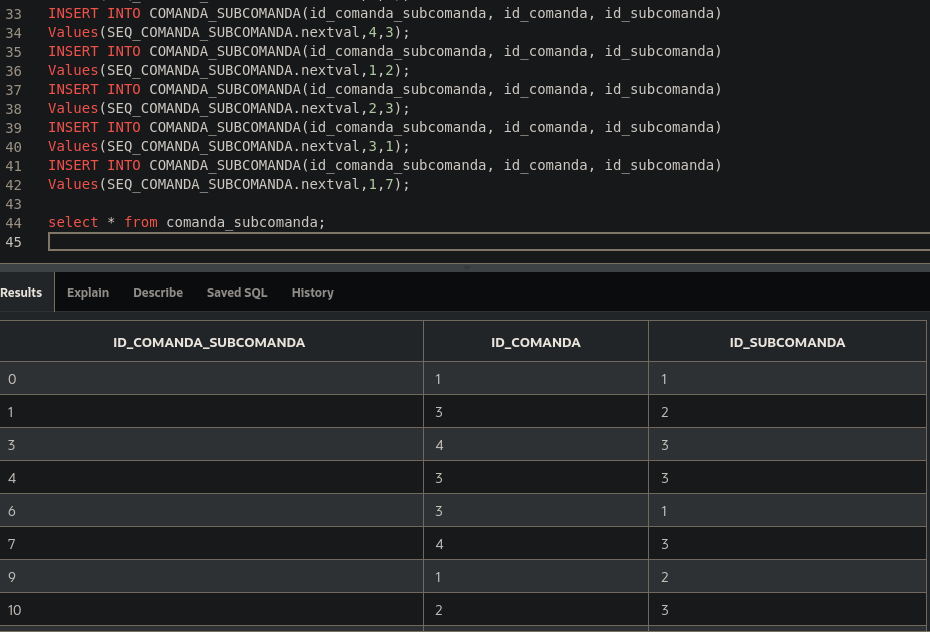
INSERT INTO COMANDA\_SUBCOMANDA(id\_comanda\_subcomanda, id\_comanda, id\_subcomanda)

Values(SEQ\_COMANDA\_SUBCOMANDA.nextval,3,1);

INSERT INTO COMANDA\_SUBCOMANDA(id\_comanda\_subcomanda, id\_comanda, id\_subcomanda)

Values(SEQ\_COMANDA\_SUBCOMANDA.nextval,1,7);

select \* from comanda\_subcomanda;



* Creare si inserare date în tabelul NOTA\_PLATA

CREATE TABLE NOTA\_PLATA

(

id\_nota\_plata number(3) constraint pk\_nota\_plata primary key,

id\_comanda number(3) not null constraint fk\_comanda references COMANDA(id\_comanda)

);

CREATE SEQUENCE SEQ\_NOTA\_PLATA

INCREMENT by 1

START WITH 0

MINVALUE 0

MAXVALUE 100

NOCYCLE

NOCACHE

ORDER;

INSERT INTO NOTA\_PLATA(id\_nota\_plata, id\_comanda)

VALUES(SEQ\_NOTA\_PLATA.nextval, 2);

INSERT INTO NOTA\_PLATA(id\_nota\_plata, id\_comanda)

VALUES(SEQ\_NOTA\_PLATA.nextval, 1);

INSERT INTO NOTA\_PLATA(id\_nota\_plata, id\_comanda)

VALUES(SEQ\_NOTA\_PLATA.nextval, 3);

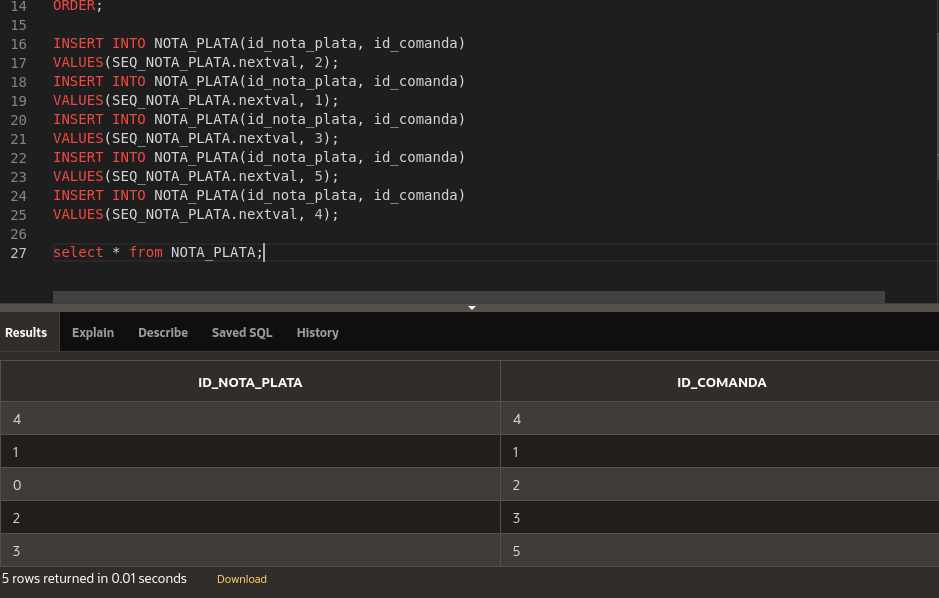
INSERT INTO NOTA\_PLATA(id\_nota\_plata, id\_comanda)

VALUES(SEQ\_NOTA\_PLATA.nextval, 5);

INSERT INTO NOTA\_PLATA(id\_nota\_plata, id\_comanda)

VALUES(SEQ\_NOTA\_PLATA.nextval, 4);

select \* from NOTA\_PLATA;



* Creare si inserare date în tabelul COMANDA\_CLIENT\_OSPATAR

CREATE TABLE COMANDA\_CLIENT\_OSPATAR

(

id\_angajat number(3) not null constraint fk\_cco\_ang references ANGAJATI(id\_angajat),

id\_client number(3) not null constraint fk\_cco\_client references CLIENTI(id\_client),

id\_comanda number(3) not null constraint fk\_cco\_comanda references COMANDA(id\_comanda),

constraint pk\_cco primary key(id\_angajat, id\_client, id\_comanda)

);

select \* from angajati;

INSERT INTO COMANDA\_CLIENT\_OSPATAR (id\_angajat, id\_client, id\_comanda)

values(9,1,1);

INSERT INTO COMANDA\_CLIENT\_OSPATAR (id\_angajat, id\_client, id\_comanda)

values(13,2,1);

INSERT INTO COMANDA\_CLIENT\_OSPATAR (id\_angajat, id\_client, id\_comanda)

values(14,3,2);

INSERT INTO COMANDA\_CLIENT\_OSPATAR (id\_angajat, id\_client, id\_comanda)

values(17,2,5);

INSERT INTO COMANDA\_CLIENT\_OSPATAR (id\_angajat, id\_client, id\_comanda)

values(16,3,2);

INSERT INTO COMANDA\_CLIENT\_OSPATAR (id\_angajat, id\_client, id\_comanda)

values(12,4,1);

INSERT INTO COMANDA\_CLIENT\_OSPATAR (id\_angajat, id\_client, id\_comanda)

values(11,2,1);

INSERT INTO COMANDA\_CLIENT\_OSPATAR (id\_angajat, id\_client, id\_comanda)

values(10,3,3);

INSERT INTO COMANDA\_CLIENT\_OSPATAR (id\_angajat, id\_client, id\_comanda)

values(9,2,3);

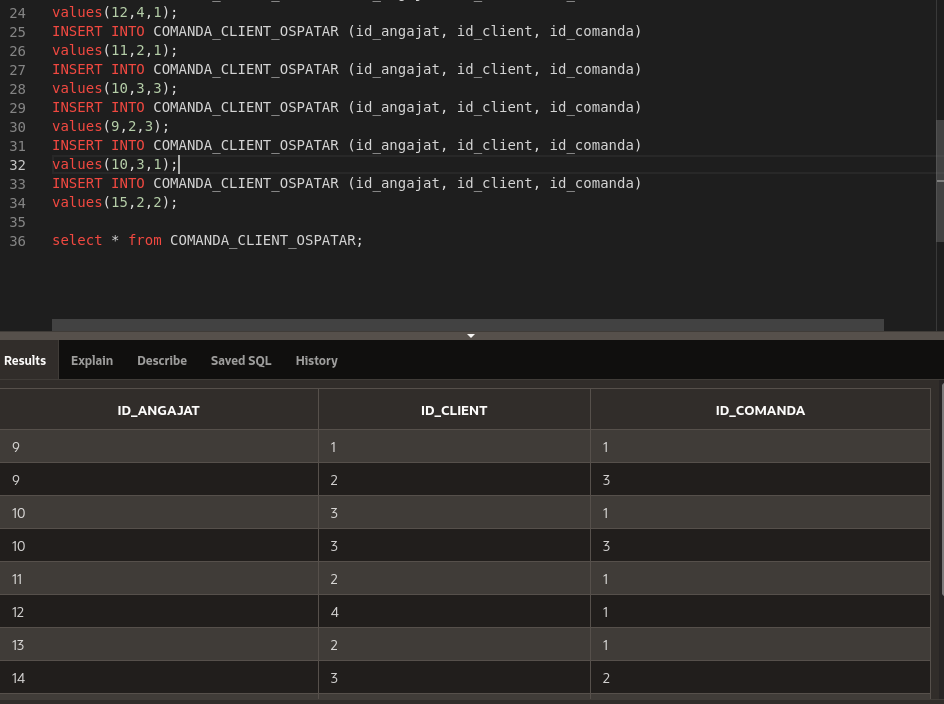
INSERT INTO COMANDA\_CLIENT\_OSPATAR (id\_angajat, id\_client, id\_comanda)

values(10,3,1);

INSERT INTO COMANDA\_CLIENT\_OSPATAR (id\_angajat, id\_client, id\_comanda)

values(15,2,2);

select \* from COMANDA\_CLIENT\_OSPATAR;



1. Formulati in limbaj natural si implementati 5 cereri SQL complexe ce vor utiliza, in ansamblul lor, următoarele elemente:

* Operatie join pe cel putin 4 tabele
* Filtrare la nivel de linii
* Subcereri sincronizate în care intervin cel puțin 3 tabele
* Subcereri nesincronizate în care intervin cel puțin 3 tabele
* Grupari de date, functii grup, filtrare la nivel de grupuri
* Ordonari
* Utilizarea a cel puțin 2 funcții pe șiruri de caractere, 2 functii pe date calendaristice, a functiilor NVL si DECODE, a cel putin unei expresii CASE
* Utilizarea a cel puțin 1 bloc de cerere (clauza WITH)

1. Sa se afișeze în ordine crescătoare clienți după numărul de comenzi efectuate.

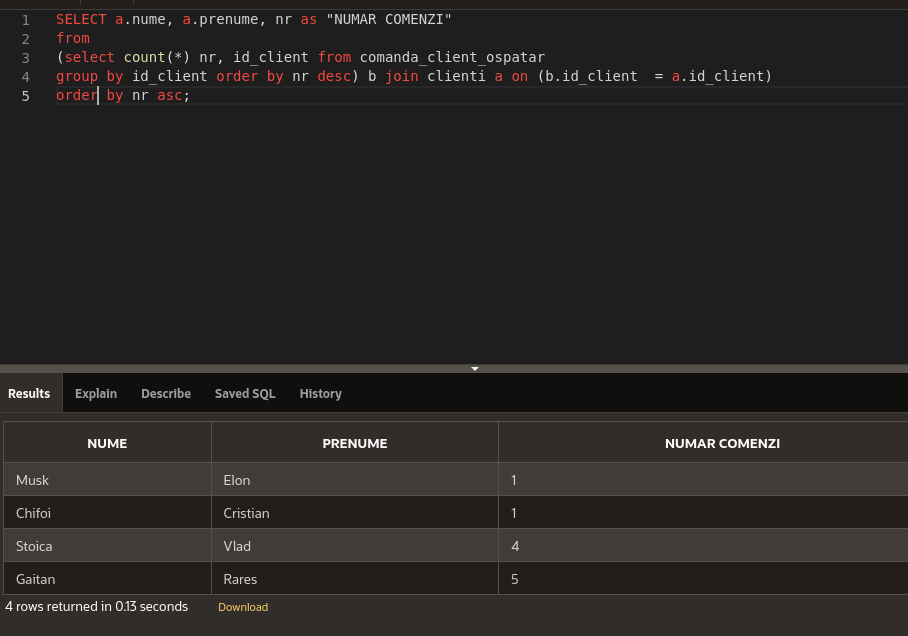
SELECT a.nume, a.prenume, nr as "NUMAR COMENZI"

from

(select count(\*) nr, id\_client from comanda\_client\_ospatar

group by id\_client order by nr desc) b join clienti a on (b.id\_client = a.id\_client)

order by nr asc;



2. Sa se afișeze numărul de comenzi pe care le-a preparat fiecare bucatar

select a.nume, a.prenume, a.tip,

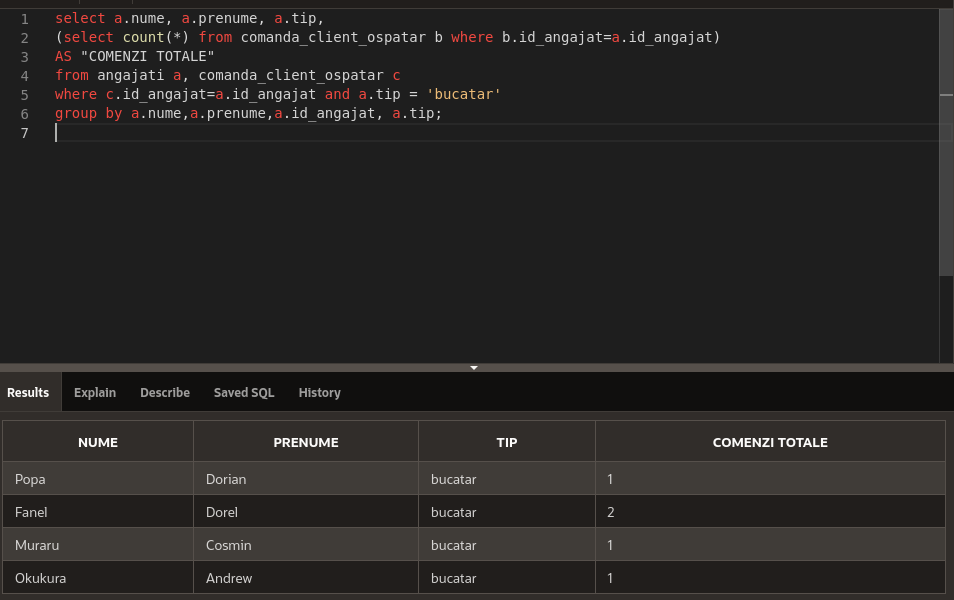
(select count(\*) from comanda\_client\_ospatar b where b.id\_angajat=a.id\_angajat)

AS "COMENZI TOTALE"

from angajati a, comanda\_client\_ospatar c

where c.id\_angajat=a.id\_angajat and a.tip = 'bucatar'

group by a.nume,a.prenume,a.id\_angajat, a.tip;



3. Sa se afișeze numele, salariul (dacă campul are valoarea null atunci se va afișa 0) și restaurantul în care lucrează ospatarii care l-au servit pe 'Gaitan Rares'

select angajat.nume NUME, NVL(angajat.salariu, 0) as salariu,

DECODE(l.id\_locatie,

1, 'Anturaj',

2,'Casa Romascana',

3, 'MGH',

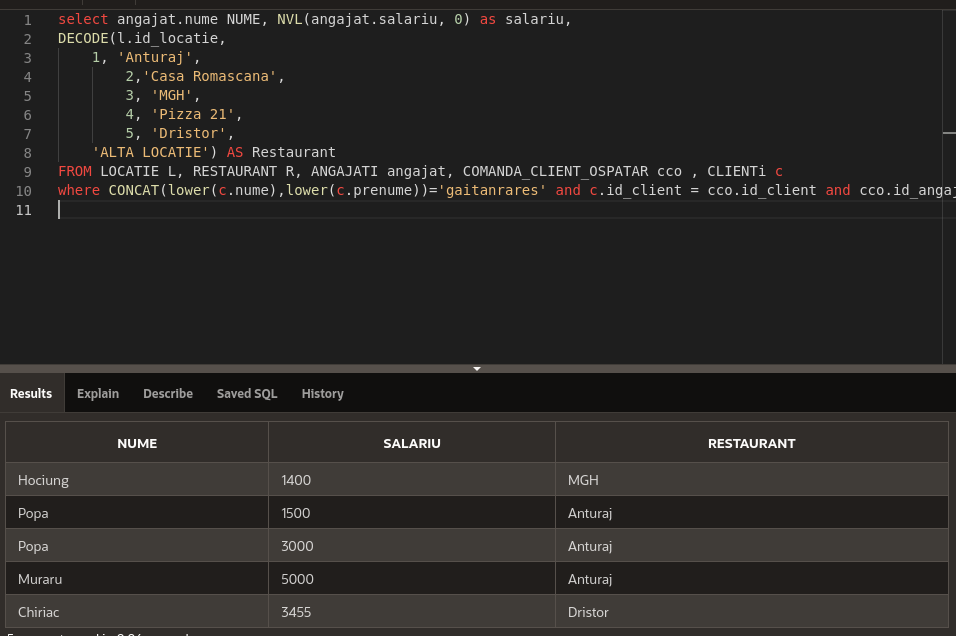
4, 'Pizza 21',

5, 'Dristor',

'ALTA LOCATIE') AS Restaurant

FROM LOCATIE L, RESTAURANT R, ANGAJATI angajat, COMANDA\_CLIENT\_OSPATAR cco , CLIENTi c

where CONCAT(lower(c.nume),lower(c.prenume))='gaitanrares' and c.id\_client = cco.id\_client and cco.id\_angajat = angajat.id\_angajat and angajat.id\_restaurant = r.id\_restaurant and r.id\_locatie = l.id\_locatie;



4. Sa se afiseze salariul total al angajaților care lucrează în restaurantul Anturaj

WITH salariu\_bucatari AS ( SELECT sum(a.salariu) suma FROM ANGAJATI a JOIN RESTAURANT r ON (a.id\_restaurant =r.id\_restaurant and r.nume = 'Anturaj') WHERE tip = 'bucatar'),

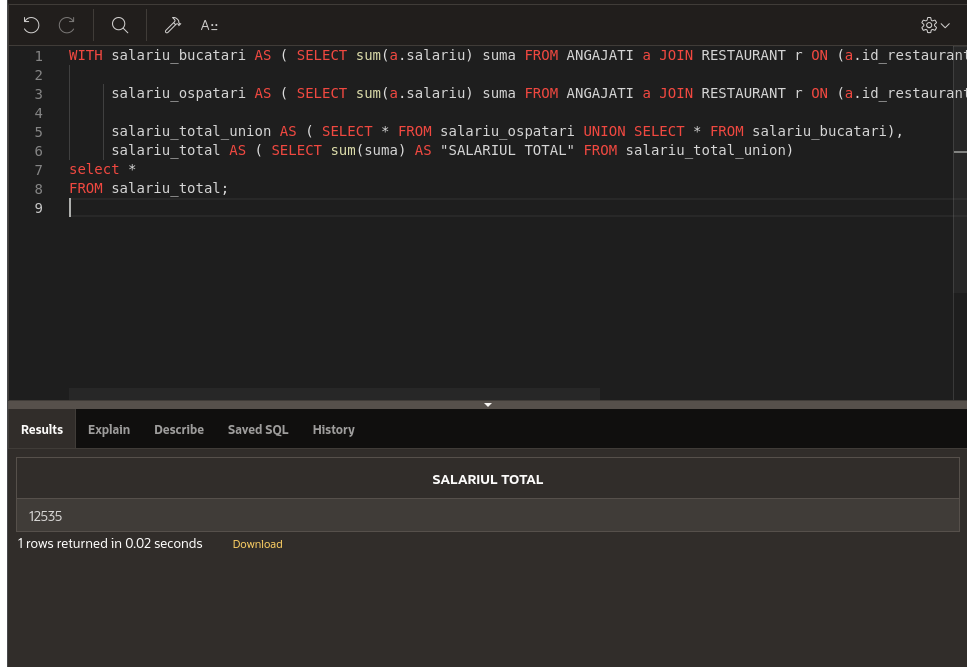
salariu\_ospatari AS ( SELECT sum(a.salariu) suma FROM ANGAJATI a JOIN RESTAURANT r ON (a.id\_restaurant =r.id\_restaurant and r.nume = 'Anturaj') WHERE tip = 'ospatar'),

salariu\_total\_union AS ( SELECT \* FROM salariu\_ospatari UNION SELECT \* FROM salariu\_bucatari),

salariu\_total AS ( SELECT sum(suma) AS "SALARIUL TOTAL" FROM salariu\_total\_union)

select \*

FROM salariu\_total;



5. Sa se afiseze locurile în care a comandat Stoica Vlad

SELECT adresa

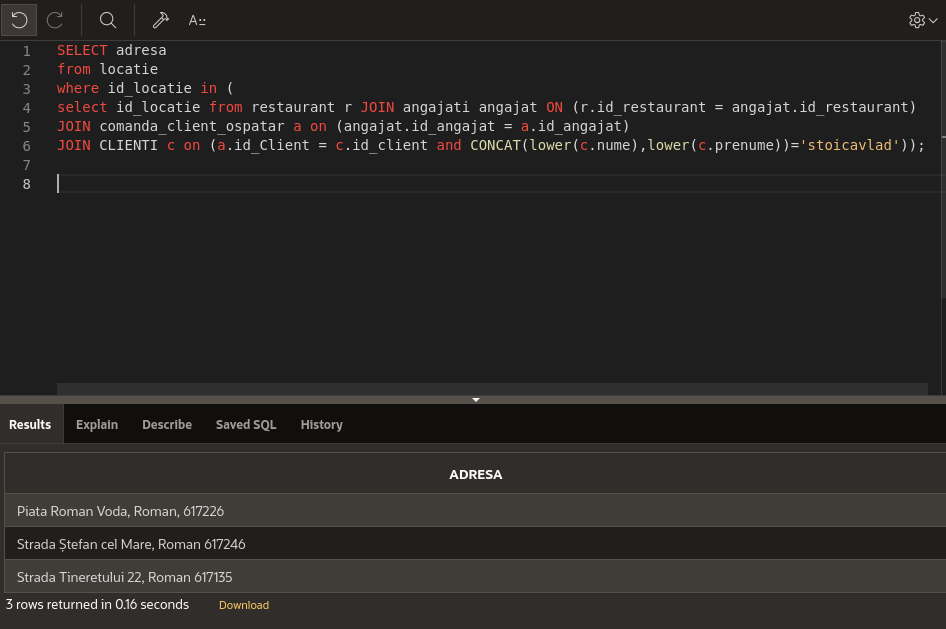
from locatie

where id\_locatie in (

select id\_locatie from restaurant r JOIN angajati angajat ON (r.id\_restaurant = angajat.id\_restaurant)

JOIN comanda\_client\_ospatar a on (angajat.id\_angajat = a.id\_angajat)

JOIN CLIENTI c on (a.id\_Client = c.id\_client and CONCAT(lower(c.nume),lower(c.prenume))='stoicavlad'));



6. Pentru fiecare angajat care este ospatar și pentru care prenumele începe cu litera M sa se afiseze datele sale și în funcție de vechime salariul să fie modificat astfel:

- dacă are mai mult de 5 ani de vechime salariul se triplează

- dacă are mai puțin de 5 ani de vechime salariul se dublează

select nume,prenume,tip,data\_angajarii, salariu,

CASE

WHEN MONTHS\_BETWEEN(SYSDATE, data\_angajarii) > 60 then salariu \* 3

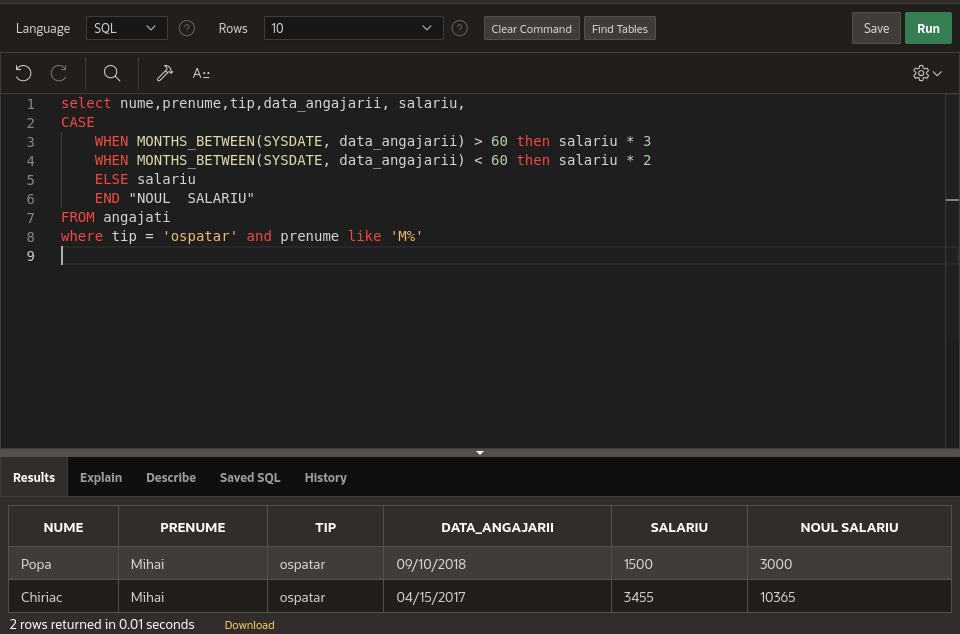
WHEN MONTHS\_BETWEEN(SYSDATE, data\_angajarii) < 60 then salariu \* 2

ELSE salariu

END "NOUL SALARIU"

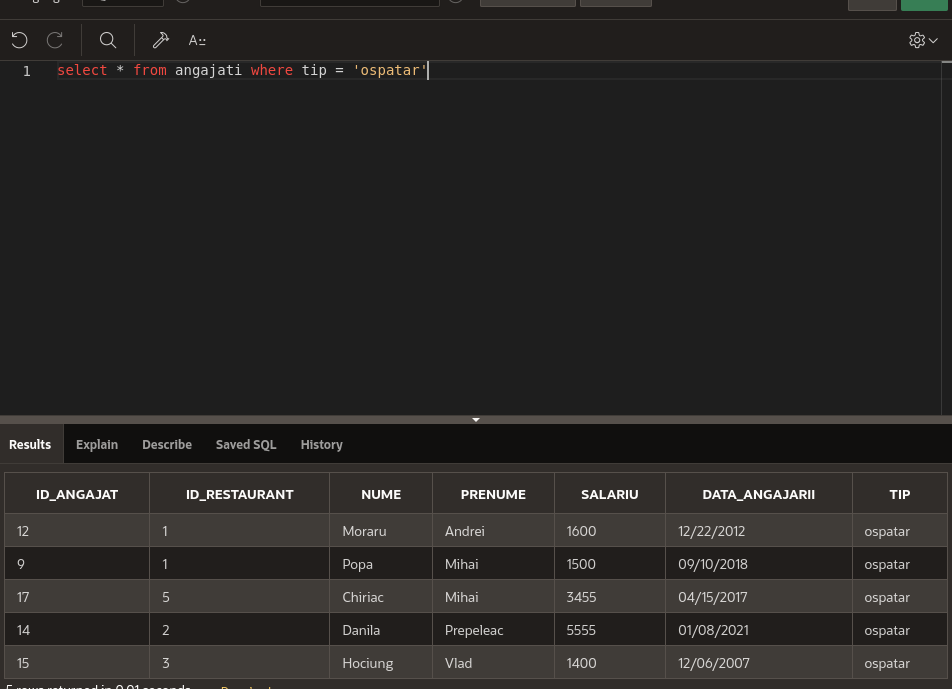
FROM angajati

where tip = 'ospatar' and prenume like 'M%'



1. Implementarea a 3 operații de actualizare sau suprimare a datelor utilizând subcereri
2. Sa se mareasca cu 2000 salariul angajatilor care au mai puțin de 3 ani de vechime.

Inainte de actualizare

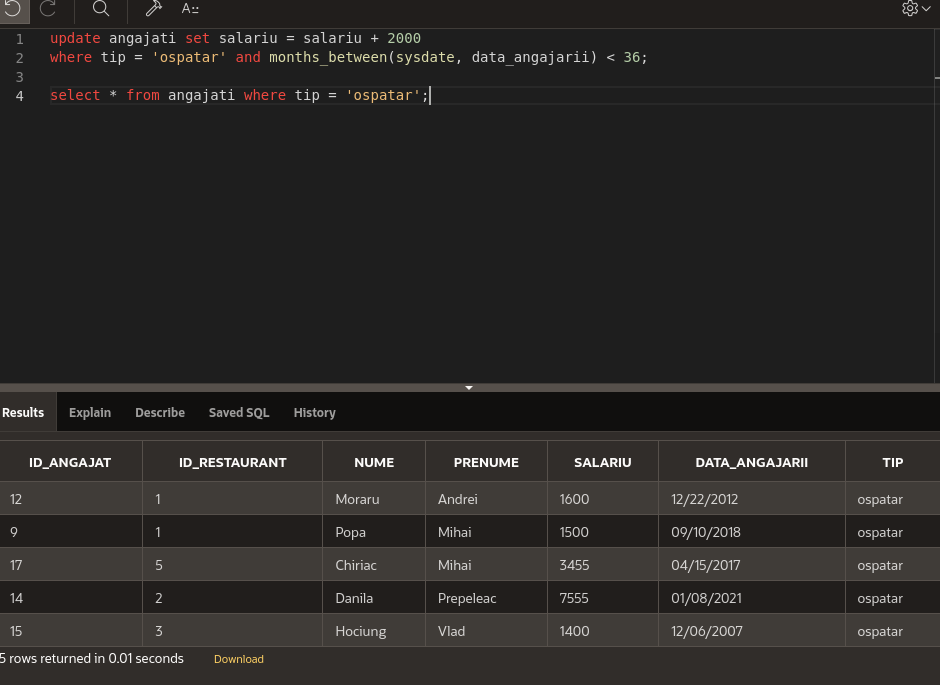


update angajati set salariu = salariu + 2000

where tip = 'ospatar' and months\_between(sysdate, data\_angajarii) < 36;

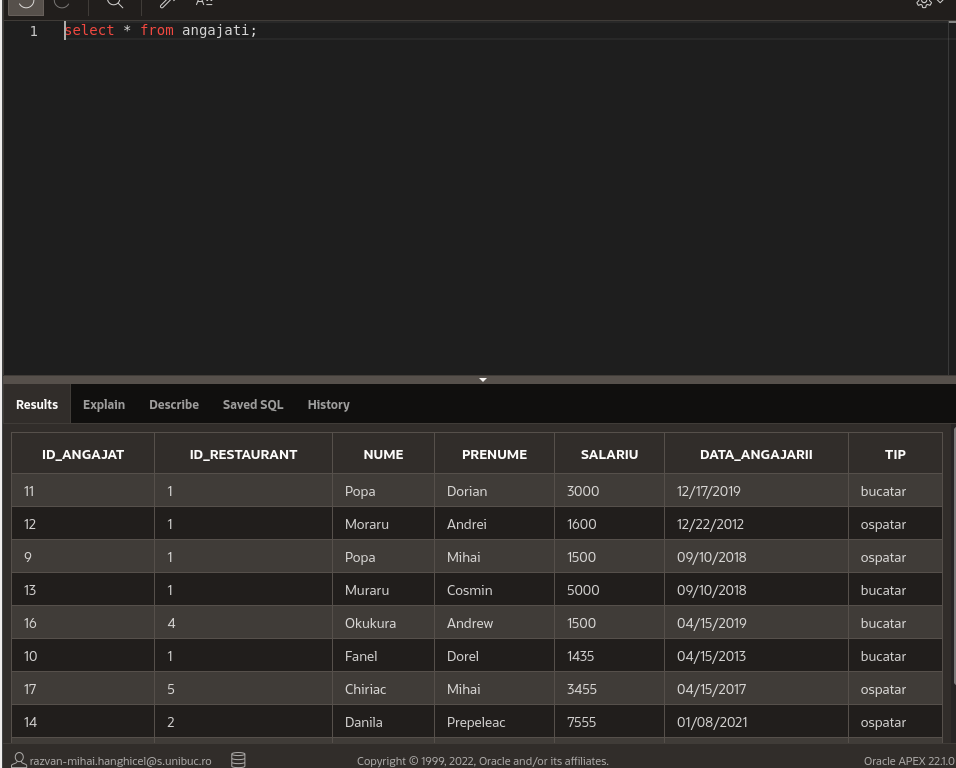
select \* from angajati where tip = 'ospatar';

Dupa actualizare



1. Sa se modifice data de angajare pentru toți angajații care au vechime mai mica de 2 ani

Inainte de stergere

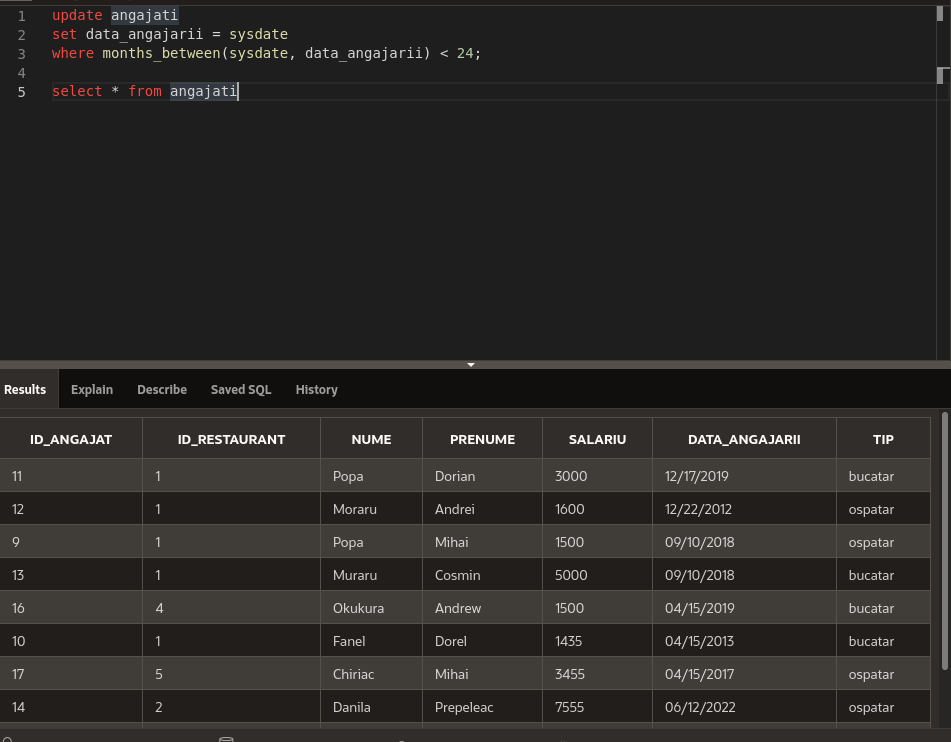


update angajati

set data\_angajarii = sysdate

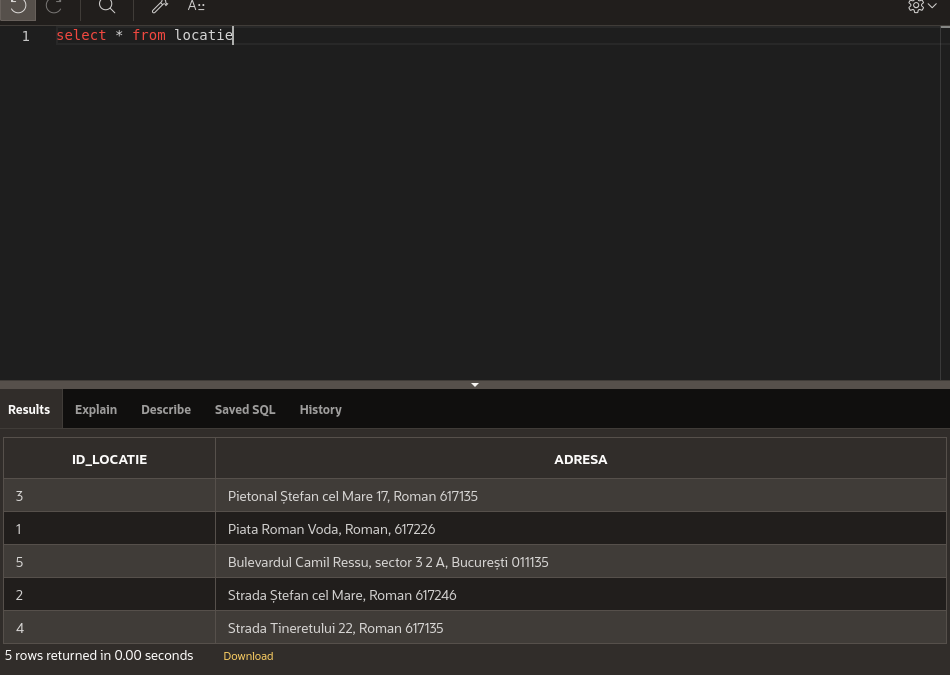
where months\_between(sysdate, data\_angajarii) < 24;

select \* from angajati



1. Modificați toate locațiile pentru care adresa nu începe cu P. Se va pune ‘nedefinita’ în locul adresei actuale

Inainte de stergere

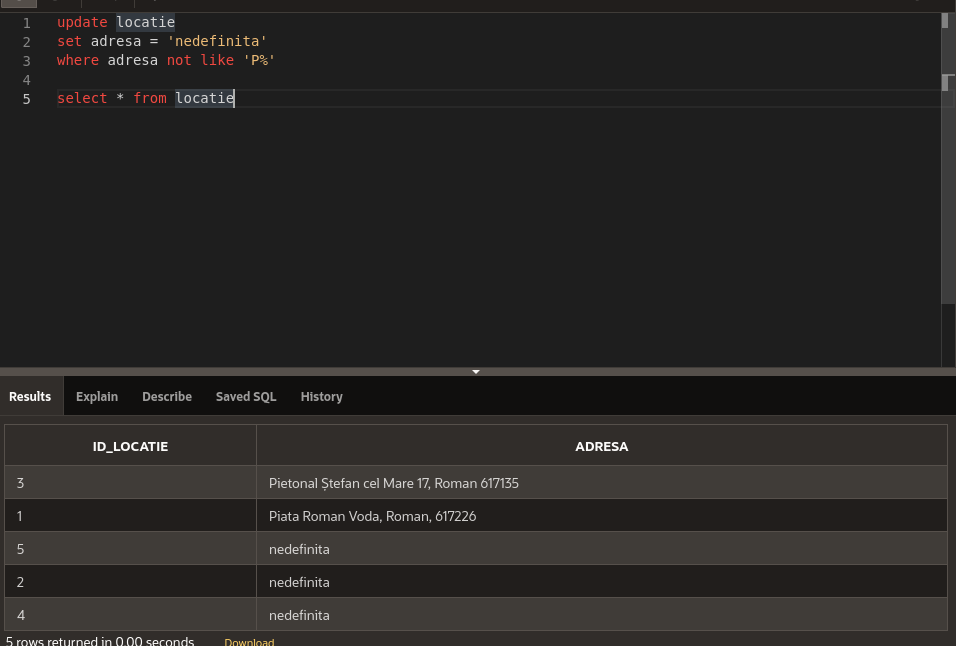


update locatie

set adresa = 'nedefinita'

where adresa not like 'P%'

select \* from locatie



1. Crearea unei secvențe ce va fi utilizate în inserarea inregistrarilor in tabele (punctul 10)

Create sequence inregistrari

Increment by 1

Start with 1

Minvalue 0

Maxvalue 100

Nocycle

Order;

1. Crearea unei vizualizari compuse. Dați un exemplu de operație LMD permisa pe vizualizarea respectivă și un exemplu de operație LMD nepermisa

create view view\_restaurant\_locatie (id\_res, nume\_res, id\_locatie, adresa) as (

select r.id\_restaurant, r.nume, l.id\_locatie, l.adresa from restaurant r, locatie l where r.id\_locatie = l.id\_locatie);

Exemplu de operatie LMD permisa:

update view\_restaurant\_locatie

set nume\_res = 'undefined'

where adresa = 'nedefinita';

Exemplu de operație LMD nepermisa:

delete from view\_restaurant\_locatie

where adresa = 'nedefinita';

1. Crearea unui index care sa optimizeze o cerere de tip căutare cu 2 criterii. Specificati cererea

CREATE TABLE rest

(id\_rest NUMBER(3) CONSTRAINT pk\_id\_rest PRIMARY KEY,

nume VARCHAR2(50) NOT NULL

)

ORGANIZATION INDEX;

CREATE TABLE loc

(id\_loc NUMBER(3) CONSTRAINT pk\_id\_loc PRIMARY KEY,

nume VARCHAR2(50) NOT NULL

);

select \* from rest

insert into rest(id\_rest, nume)

values(1, 'restaurantul 1');

insert into rest(id\_rest, nume)

values(2, 'restaurantul 2');

insert into rest(id\_rest, nume)

values(3, 'restaurantul 3');

insert into rest(id\_rest, nume)

values(4, 'restaurantul 4');

select rowid, a.\* from rest a;

1. Formulati in limbaj natural si implementati in SQL: o cerere ce utilizează operația outer-join pe minim 4 tabele și două cereri ce utilizează operația division
2. Optimizarea unei cereri, aplicând regulile de optimizare ce deriva din proprietatile operatorilor algebrei relaționale. Cererea va fi exprimata prin expresie algebrica, arbore algebric si limbaj (SQL), atat anterior cat si ulterior optimizarii

Sa se afiseze toti ospatarii care l-au servit

* + Realizarea normalizarii BCNF, FN4, FN5

**BCNF (Boyce-Codd Normal Form)**

Formal, o relație R este în BCNF dacă și numai data pentru orice dependenta functionala totala X -> A, X este o cheie candidat a lui R.

O relație R este în BCNF dacă și numai dacă fiecare determinant este o cheie candidat

O comanda poate fi cerută de mai mulți clienți, deci in acest caz avem nevoide de o cheie primara compusa. Sa presupunem ca un ospatar poate servi un singur client.

| COMANDA | CLIENT | OSPATAR |
| --- | --- | --- |
| 1 | 1 | 1 |
| 1 | 3 | 2 |
| 3 | 1 | 1 |
| 4 | 2 | 2 |

Se poate observa ca:

* Comanda 1 este ceruta de clientii 1 si 3, aceștia fiind serviti de ospatarii 1 și 2.
* Clientul 1 a plasat mai multe comenzi

{Comanda, Client} -> {Ospatar}

{Ospatar} -> {Client}, deoarece un ospatar va servi un singur client, deci el va fi o cheie candidat, aparand o singura data alături de clientul pe care-l serveste.

Vom aplica regula Casey-Delobel:

T1: (Ospatar, Comanda)

T2: (Ospatar, Client)

**Normalizare FN4**

Prin aplicarea FN4 o sa eliminam redundantele datorate relatiilor many to many.

* O relație R este în FN4 dacă și numai dacă relația este în BCNF si nu contine relatii many to many independente

De exemplu, o comanda este adusă de mai mulți ospatari, iar o comanda poate fi cerută de mai mulți clienți

| COMANDA | CLIENT | OSPATAR |
| --- | --- | --- |
| COMANDA 3 | CLIENT 1 | OSPATAR 2 |
| COMANDA 3 | CLIENT 2 | OSPATAR 1 |

O sa avem:

Comanda -> Client

Comanda -> Ospatar

Relația de mai sus nu se afla în FN4, deoarece avem multidependențele prezentate mai sus. Pentru asta trebuie sa normalizam in forma normala 4:

(Comanda, Client)

| COMANDA | CLIENT |
| --- | --- |
| COMANDA 3 | CLIENT 1 |
| COMANDA 3 | CLIENT 2 |

(Comanda, Ospatar)

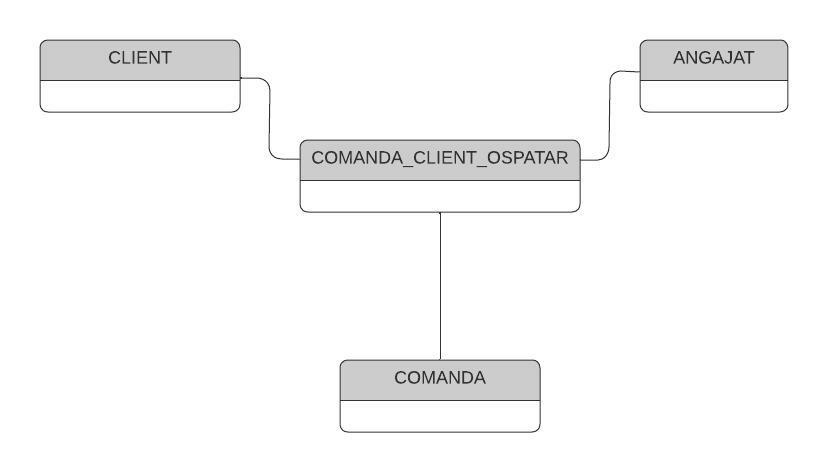
| COMANDA | OSPATAR |
| --- | --- |
| COMANDA 3 | OSPATAR 2 |
| COMANDA 3 | OSPATAR 1 |

**Normalizare FN5**

Prin aplicarea FN5 o sa eliminăm redundantele din relatiile many to many dependente. Putem spune ca o relație se afla in FN5 dacă:

* Relatia se afla in FN4
* Relatia nu contine dependente ciclice

Vom avea o relatie de tip 3, în care o sa știm exact cine a plasat o anumită comandă, cine a servit o anumită comandă și din ce este compusă o anumită comandă



* + Aplicarea denormalizarii, justificand necesitatea acesteia

Aplicând normalizarea putem elimina redundante utilizând anumite proiecții, doar ca prin aceasta metoda nu o sa putem elimina chiar toate redundantele.

Prin denormalizare intelegem:

* Reducerea de join-uri între tabele, de unde rezulta un timp de execuție mai mic
* Marirea redundantei

Sa luăm de exemplu entitatea SUBCOMANDA, in care avem un camp **cantitate**. Pe aceasta coloana putem regasi valori duplicate pentru mai multe subcomenzi. Daca in baza noastra de date am putea crea un tabel în care sa asignam fiecărei cantități un id al subcomenzii care ii corespunde, va fi nevoie de denormalizare, deoarece câmpul cantitate trebuie sa facă parte din entitatea SUBCOMANDA pentru ca nu ar fi eficient sa fie în alt tabel.