

PROJECT

Baze de date

RESTAURANT

Guțu Ștefania-Alexandra
grupa 141

CUPRINS

- 1.** Descrierea modelului real și a regulilor de funcționare
- 2.** Constrângeri
- 3.** Entități
- 4.** Relații
- 5.** Atribute
- 6.** Diagrama entitate-relație
- 7.** Diagrama conceptuală
- 8.** Scheme relaționale
- 9.** FN1, FN2, FN3
- 10.** Crearea tabelelor în SQL și inserarea datelor
 - + **13.** Crearea unei secvențe
- 11.** Cereri SQL
- 12.** Operații de actualizare și suprimare a datelor
- 16.** Cerere ce utilizează operația outer-join + cereri ce utilizează operația division
- 17.** Optimizarea unei cereri
- 18.** BCNF, FN4, FN5.

1. Descrierea modelului real și a regulilor de funcționare

Proiectul cuprinde proiectarea unei baze de date ce furnizează informații despre un restaurant, permitând o bună gestionare a comenziilor, clienților, evenimentelor, angajaților, produselor și a furnizorilor.

Tema aleasă are ca scop gestiunea tuturor datelor utile ce au în centru *comenziile* date, pentru o administrare și o organizare eficientă a restaurantului.

Regulile de funcționare:

- Există furnizori prin intermediul cărora se procură o diversitate de ingredientele necesare în pregătirea produselor. Fiecare produs face parte dintr-o categorie, exact ca într-un meniu. Categoriile pot fi de exemplu: paste, salate, pizza, băuturi, etc.
- Clienții pot solicita diferite comenzi care conțin unul sau mai multe produse din categoriile existente. Fiecare comandă poate fi plătită într-un anumit mod de plată (cash, card, etc).
- De asemenea, în cadrul restaurantului pot fi organizate evenimente de diverse tipuri: nuntă, botez, aniversări, etc. Pentru acestea se pot da una sau mai multe comenzi.
- Există angajați care pot ocupa postul de bucătar sau de ospătar. Bucătarii se ocupă cu gătirea produselor solicitate de clienți, iar ospătarii se ocupă cu servirea acestora.
- Produsele conținute de comanda unui client pot fi gătite de bucătari diferiți și servite de mai mulți ospătari.

2. Constrângeri

Modelul de date respectă anumite restricții de funcționare:

- Un eveniment poate fi de un singur tip. Pentru un eveniment trebuie să se dea cel puțin o comandă, dar se pot da și mai multe comenzi.
- O comandă trebuie să fie solicitată de un client și trebuie plătită într-un mod de plată.
- Un ingredient trebuie să fie adus de un furnizor. Doi furnizori sau mai mulți nu pot aduce același tip de ingredient.
- Un produs trebuie să facă parte dintr-o categorie.
- Un ingredient poate să nu fie folosit la prepararea produselor, dar un produs trebuie să conțină cel puțin un ingredient.
- În cadrul unei comenzi, pot exista mai mulți bucătari care să gătească același tip de produs, în schimb, un bucătar gătește singur un anumit produs (nu este ajutat de ceilalți).

3. Entități

Pentru modelul de date referitor la prezentările de modă, structurile COMANDA, CLIENT, EVENIMENT, TIP_EVENIMENT, MOD_DE_PLATA, ANGAJAT, BUCATAR, OSPATAR, CATEGORIE, PRODUS, INGREDIENT, FURNIZOR reprezintă entități.

Toate entitățile care vor fi prezentate sunt independente, cu excepția subentităților BUCATAR și OSPATAR.

COMANDA = cerere prin care o persoană fizică solicită un produs sau mai multe pentru a fi realizate și servite de angajații restaurantului. Aceasta poate fi plătită în mai multe moduri de plată. Cheia primară a entității este *id_comandă*.

CLIENT = persoană fizică care solicita comenzi pentru a consuma produse sau care organizează evenimente în cadrul restaurantului. Această entitate are cheia primară *id_client*.

EVENIMENT = fapt de o importanță deosebită pentru client care dorește organizarea evenimentului în cadrul restaurantului. Acesta poate fi de mai multe tipuri în funcție de întâmplarea deosebită din viața clientului. Atributul *id_eveniment* reprezintă cheia primară a acestei entități.

TIP_EVENIMENT = particularitatea evenimentului stabilită în funcție de întâmplarea importantă din viața clientului (exemplu: nuntă, botez, aniversare, etc). Cheia primară a entității este *id_tip_eveniment*.

MOD_DE_PLATA = modalitatea prin care clientul achită nota comenzi plasate în cadrul restaurantului. Atributul *id_mod_plata* este cheia primară a acestei entități.

ANGAJAT = persoană fizică care lucrează în restaurant pe baza unui contract de muncă. Această entitate are cheia primară *id_angajat*.

BUCATAR = subentitate a entității ANGAJAT, ce conține informații specifice bucătarilor. Cheia primară a entității este *id_angajat*.

OSPATAR = subentitate a entității ANGAJAT, ce conține informații specifice ospătarilor. Cheia primară a entității este *id_angajat*.

CATEGORIE = grup de produse asemănătoare între ele(exemplu: supe, pizza, salate, bauturi, etc). Această entitate are cheia primară *id_categorie*.

PRODUS = preparat culinar obținut printr-un amestec de ingrediente. Produsele pot fi împărțite în mai multe categorii. Atributul *id_produs* reprezintă cheia primară a acestei entități.

INGREDIENT = substanță care intră în compoziția unui produs și care este furnizat de un furnizor. Această entitate are cheia primară *id_ingredient*.

FURNIZOR = persoană fizică sau juridică care furnizează diferite ingrediente necesare în prepararea produselor. Cheia primară a acestei entități este *id_furnizor*.

4. Relatii

CLIENT_cere_COMANDA = relație care leagă entitățile CLIENT și COMANDA, reflectând legătura dintre acestea (ce comandă cere un anumit client). Ea are cardinalitatea minimă 1:1 (un client trebuie să ceară cel puțin o comandă, iar o comandă trebuie să fie cerută de un client) și cardinalitatea maximă 1:n (un client poate cere mai multe comenzi, iar o comandă poate fi cerută de un singur client).

COMANDA_facuta_pentru_EVENTIMENT = relație care leagă entitățile COMANDA și EVENTIMENT, reflectând legătura dintre acestea (ce comenzi sunt făcute pentru un anumit eveniment). Relația are cardinalitatea minimă 1:0 (o comandă nu trebuie neapărat să fie facută pentru un eveniment, iar pentru un eveniment trebuie să se dea cel puțin o comandă) și cardinalitatea maximă n:1 (o comanda poate fi facută pentru un singur eveniment, iar pentru un eveniment se pot da mai multe comenzi).

EVENTIMENT_este_TIP_EVENTIMENT = relație care leagă entitățile EVENTIMENT și TIP_EVENTIMENT, reflectând legătura dintre acestea (ce tip de eveniment este un anumit eveniment). Ea are cardinalitatea minimă 0:1 (un eveniment trebuie să fie de un anumit tip, dar nu trebuie să fie neapărat un eveniment cu un anumit tip) și cardinalitatea maximă n:1 (un eveniment poate să fie de un singur tip, iar tipurile de evenimente pot apartine mai multor evenimente).

COMANDA_platita_MOD_DE_PLATA = relație care leagă entitățile COMANDA și MOD_DE_PLATA, reflectând legătura dintre acestea (prin ce mod de plată a fost plătită o comandă). Relația are cardinalitatea minimă 0:1 (o comandă trebuie să fie plătită într-un anumit mod, dar nu trebuie să fie neapărat o comandă plătită într-un anumit mod) și cardinalitatea maximă n:1 (o comandă poate fi plătită într-un singur mod, iar în același mod de plată pot fi plătite mai multe comenzi).

ANGAJAT_gateste/serveste_PRODUS_din_COMANDA = relație de tip 3 ce leagă entitățile ANGAJAT, PRODUS și COMANDA, reflectând ce angajat a gătit/servit un anumit produs și pentru ce comandă. Denumirea acestei relații va fi *gateste/serveste*.

PRODUS_face_parte_CATEGORIE = relație care leagă entitățile PRODUS și CATEGORIE, reflectând legătura dintre acestea (din ce categorie face parte un produs). Ea are cardinalitatea minimă 1:1 (dintr-o categorie trebuie să facă parte cel puțin un produs, iar un produs trebuie să facă parte dintr-o categorie) și cardinalitatea maximă 1:n (dintr-o

categorie pot face parte mai multe produse, iar un produs poate face parte dintr-o singura categorie).

PRODUS_contine_INGREDIENT = relație care leagă entitățile PRODUS și INGREDIENT, reflectând legătura dintre acestea (ce ingrediente conține un produs). Relația are cardinalitatea minimă 0:1 (un produs trebuie să conțină cel puțin un ingredient, iar un ingredient nu trebuie să fie conținut neapărat de un produs) și cardinalitatea maximă m:n (un produs poate conține mai multe ingrediente, iar un ingredient poate fi conținut de mai multe produse).

INGREDIENT_adus_de_FURNIZOR = relație care leagă entitățile INGREDIENT și FURNIZOR, reflectând legătura dintre acestea (ce ingrediente aduce un anumit furnizor). Ea are cardinalitatea minimă 0:1 (un ingredient trebuie să fie adus de un furnizor, iar un furnizor nu trebuie să aducă neapărat un ingredient) și cardinalitatea maximă n:1 (un ingredient poate fi adus de un singur furnizor, iar un furnizor poate aduce mai multe ingrediente).

5. **Atribute**

Entitatea *CLIENT* are ca atribute:

id_client = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul unui client.

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

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

nr_telefon = variabilă de tip caracter, de lungime 10, care reprezintă numarul de telefon al clientului.

Entitatea *COMANDA* are ca atribute:

id_comanda = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul unei comenzi.

data_comanda = variabilă de tip dată calendaristică, care reprezintă data la care a fost plasată comanda.

id_mod_plata = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul unui mod de plată. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul MOD_PLATA.

id_client = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul clientului care a plasat comanda. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul CLIENT.

id_eveniment = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul unui eveniment. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul EVENIMENT.

Entitatea *EVENIMENT* are ca atribute:

id_eveniment = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul unui eveniment.

data_eveniment = variabilă de tip dată calendaristică, care reprezintă data la care a avut loc evenimentul.

preferinte = variabila de tip sir de caractere de lungime variabilă (care nu ocupă mai mult de 2G), care include preferințele clientului.

id_tip_eveniment = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul unei tip de eveniment. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul TIP_EVENIMENT.

Entitatea TIP_EVENIMENT are ca atribute:

id_tip_eveniment = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul unei tip de eveniment.

denumire_tip = variabilă de tip caracter, de lungime maximă 20, care reprezintă denumirea tipului de eveniment. De exemplu, poate lua valorile nuntă, botez, aniversare, cununie, revedere, etc.

descriere = variabila de tip sir de caractere de lungime variabilă (care nu ocupă mai mult de 2G), care include descrierea tipului de eveniment.

Entitatea MOD_DE_PLATA are ca atribute:

id_mod_plata = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul unui mod de plată.

denumire = variabilă de tip caracter, de lungime maximă 20, care reprezintă denumirea modului de plată al comenzii. Atributul poate lua valorile cash, card, online, etc.

observatii = variabila de tip sir de caractere de lungime variabilă (care nu ocupă mai mult de 2G), care include observații despre modul de plata.

Entitatea ANGAJAT are ca atribute:

id_angajat = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul unui angajat.

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

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

email = variabilă de tip caracter, de lungime maximă 50, care reprezintă email-ul angajatului.

nr_telefon = variabilă de tip caracter, de lungime 10, care reprezintă numarul de telefon al angajatului.

data_angajare = variabilă de tip dată calendaristică, care reprezintă data la care a fost angajată persoana respectivă.

tip_angajat = variabilă de tip caracter, de lungime maxima 10, care reprezintă funcția pentru care a fost angajată persoana respectivă. De exemplu, poate să fie bucătar sau ospătar.

Subentitatea BUCATAR are ca atribute:

id_angajat = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul unui bucătar.

specializare = variabilă de tip caracter, de lungime maximă 50, care reprezintă specializarea bucătarului.

Subentitatea *OSPATAR* are ca atribute:

id_angajat = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul unui ospătar.

experienta = variabilă de tip întreg, de lungime maximă 2, care reprezintă anii de experiență pe care îi are ospătarul.

Entitatea *CATEGORIE* are ca atribute:

id_categorie = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul unei categorii.

nume_categorie = variabilă de tip caracter, de lungime maximă 20, care reprezintă numele categoriei. De exemplu, acest atribut poate lua valorile supe, pizza, salate, bauturi, etc.

Entitatea *PRODUS* are ca atribute:

id_produs = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul unui produs.

denumire = variabilă de tip caracter, de lungime maximă 30, care reprezintă denumirea produsului.

pret = variabilă de tip întreg, de lungime maximă 3, care reprezintă prețul unui produs.

observatii = variabila de tip sir de caractere de lungime variabilă (care nu ocupă mai mult de 2G), care include observații despre produs.

id_categorie = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul unei categorii. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul *CATEGORIE*.

Entitatea *INGREDIENT* are ca atribute:

id_ingredient = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul unui ingredient.

denumire = variabilă de tip caracter, de lungime maximă 20, care reprezintă denumirea ingredientului.

id_furnizor = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul unui furnizor. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul *FURNIZOR*.

Entitatea *FURNIZOR* are ca atribute:

id_furnizor = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul unui furnizor.

denumire = variabilă de tip caracter, de lungime maximă 20, care reprezintă denumirea furnizorului.

nr_telefon = variabilă de tip caracter, de lungime 10, care reprezintă numarul de telefon al furnizorului.

Relația *PRODUS_contine_INGREDIENT* are ca atribute:

id_produs = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul unui produs. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul *PRODUS*.

id_ingredient = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul unui ingredient. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul *INGREDIENT*.

Relația *ANGAJAT_gateste/serveste_PRODUS_din_COMANDA* are ca atrbute:

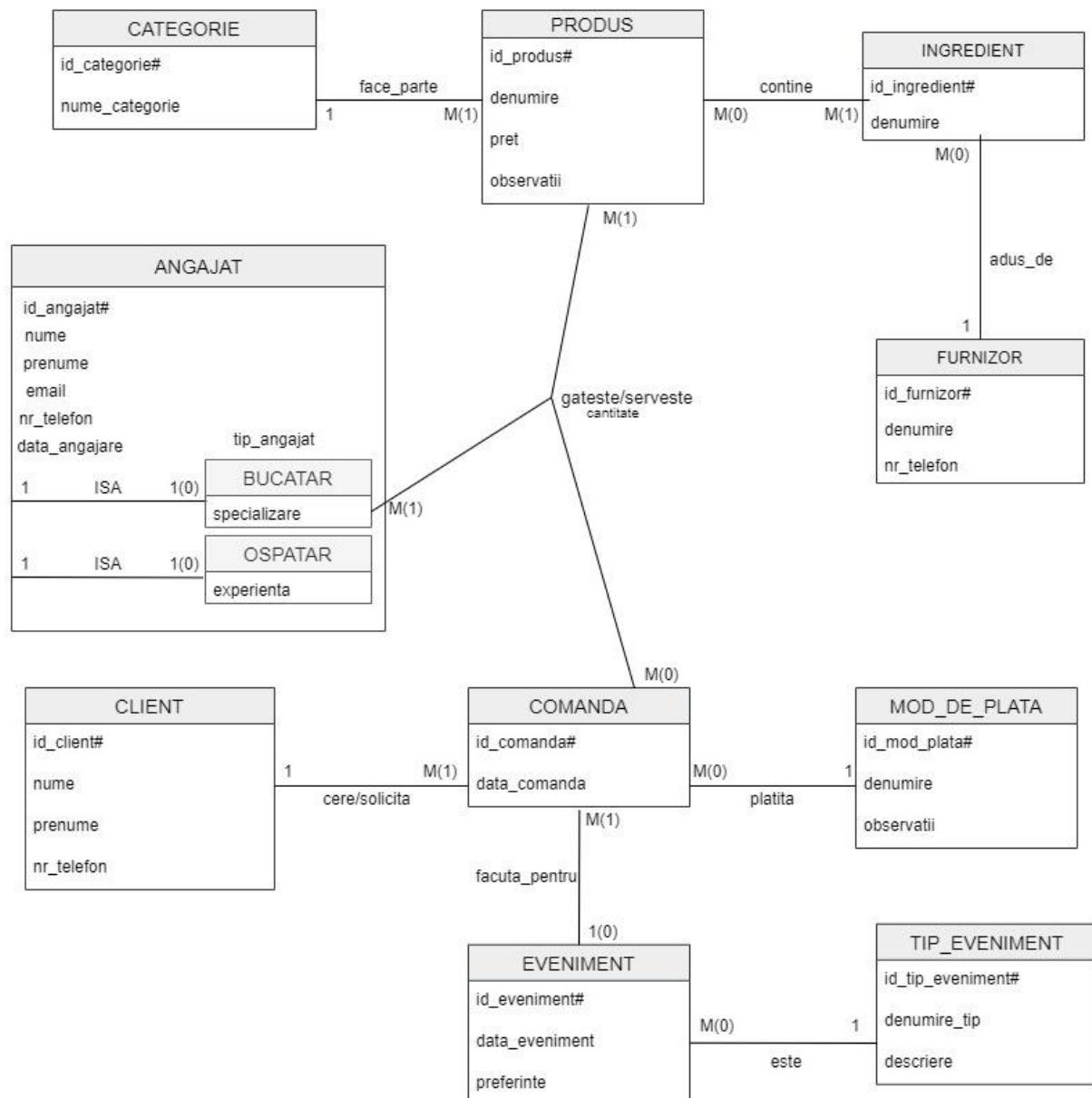
id_angajat = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul unui angajat. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul ANGAJAT.

id_produs = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul unui produs. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul PRODUS.

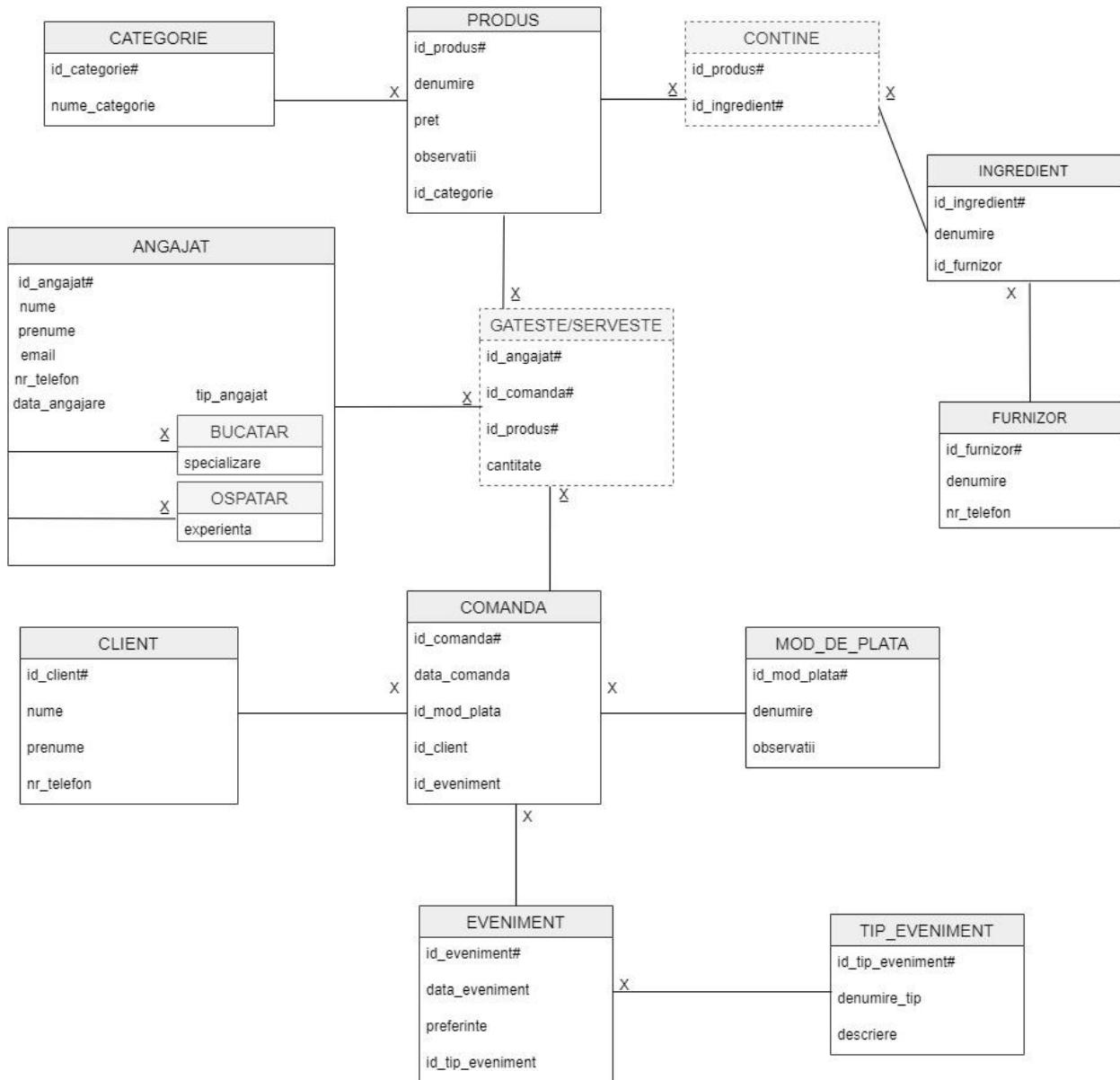
id_comanda = variabilă de tip întreg, de lungime maximă 5, care reprezintă codul unei comenzi. Atributul trebuie să corespundă la o valoare a cheii primare din tabelul COMANDA.

cantitate = variabilă de tip întreg, de lungime maximă 2, care reprezintă cantitatea de produse de un anumit tip care este gătită pentru comanda respectivă.

6. *Diagrama entitate-relație*



7. Diagrama conceptuală



8. Scheme relationale

Schemele relationale corespunzătoare diagramei conceptuale sunt următoarele:

- COMANDA(id_comanda#, data_comanda, id_mod_plata, id_client, id_eveniment)
- CLIENT(id_client#, nume, prenume, nr_telefon)
- EVENIMENT(id_eveniment#, data_eveniment, preferinte, id_tip_eveniment)
- TIP_EVENIMENT(id_tip_eveniment#, denumire_tip, descriere)
- MOD_DE_PLATA(id_mod_plata#, denumire, observatii)
- ANAJAT(id_angajat#, nume, prenume, email, nr_telefon, data_angajare, tip_angajat)
- BUCATAR(id_angajat#, specializare)
- OSPATAR(id_angajat#, experienta)

GATESTE/SERVESTE(id_angajat#, id_produs#, id_comanda#, cantitate)
 CATEGORIE(id_categorie#, nume_categorie)
 PRODUS(id_produs#, denumire, pret, observatii, id_categorie)
 CONTINE(id_produs#, id_ingredient#)
 INGREDIENT(id_ingredient#, denumire, id_furnizor)
 FURNIZOR(id_furnizor#, denumire, nr_telefon)

9. **FN1, FN2, FN3**

FN1

Ca o relație să fie în prima formă normală trebuie ca fiecărui atribut care o compune să îi corespundă o valoare indivizibilă.

Inițial tabelul asociativ CONTINE arăta în felul următor:

CONTINE	
id_produs#	id_ingredient
54	179,171,173
53	174,176,177,178

Produsele cu id-urile 53 și 54 se gasesc în tabelul PRODUSE, iar ingredientele cu id-urile 179, 171, 173, 174, 176, 177, 178 se gasesc în tabelul INGREDIENTE. Un produs poate să conțină unul sau mai multe ingrediente, iar un ingredient poate fi conținut de mai multe produse.

Pizza margherita (54) conține: faina(179), sos de rosii(171), parmezan(173).

Salata greceasca (53) conține: salata verde(174),rosii(176),castraveti(177), branza(178).

După realizarea normalizării în FN1, tabelul CONTINE arată în felul următor:

CONTINE	
id_produs#	id_ingredient#
54	179
54	171
54	173
53	174
53	176
53	177
53	178

Astfel, fiecare atribut are o cheie primară unică, iar, acum, relația se află în FN1.

FN2

Ca o relație să fie în a doua formă normală trebuie ca fiecare atribut care nu este cheie să fie dependent de întreaga cheie primară. (+ să fie în FN1)

Initial tabelul asociativ GATESTE\SERVESTE arăta în felul următor:

GATESTE\SERVESTE				
id_angajat#	id_comanda#	id_produs#	denumire	cantitate
204	10	57	clatite	8
206	12	52	salata cu pui	2
205	11	57	clatite	3

Un angajat gătește/servește anumite produse din diferite comenzi și poate să gătească/servească mai multe tipuri de produse. În cazul dat, atributele *denumire* și *cantitate* nu fac parte din cheie, deci acestea trebuie să fie dependente de cheia primară. Cantitatea este în strânsă legătură cu cele 3 id-uri, deoarece se referă strict la cantitatea dintr-un anumit produs pe care o gătește/servește un angajat pentru o anumită comandă. Însă, se observă că denumirea nu depinde direct de întreaga cheie primară, aceasta depinde direct de *id_produs*, înseamnă că denumire depinde numai de o parte a cheii primare și anume de *id_produs* => relația gateste/serveste nu este în FN2

{*id_produs#*} -> {*denumire*}
{*id_angajat#, id_comanda#, id_produs#*} -> {*cantitate*}
=> Două proiecții: GATESTE/SERVESTE(*id_angajat#, id_comanda#, id_produs#, cantitate*)
PRODUSE(*id_produs#, denumire*)

După realizarea normalizării în FN2 :

GATESTE\SERVESTE			
id_angajat#	id_comanda#	id_produs#	cantitate
204	10	57	8
206	12	52	2
205	11	57	3

PRODUSE	
id_produs#	denumire
57	clatite
52	salata cu pui

Astfel, fiecare atribut care nu este cheie este dependent de întreaga cheie primară, iar, acum, relația se află în FN2.

FN3

Că o relație să fie în a treia formă normală trebuie ca fiecare atribut care nu este cheie să depindă direct de cheia primară. (+ să fie în FN2)

Initial tabelul EVENIMENT arăta în felul următor:

EVENIMENT			
id_eveniment#	data_eveniment	denumire_tip	descriere_tip
110	13-07-2021	nunta	maxim 150 pers
111	25-06-2021	revedere	liceu/facultate
112	15-09-2020	cununie	null
113	03-04-2021	revedere	liceu/facultate

În acest caz, am observat că descriere_tip depinde tranzitiv de cheia primară prin intermediul atributului denumire_tip.

{id_eveniment#} -> {data_eveniment, denumire_tip} - id_eveniment determină funcțional attributele data_eveniment și denumire_tip
{id_eveniment#} -> {data_eveniment, denumire_tip} -> {descriere_tip}

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

EVENIMENT(id_eveniment#, data_eveniment, denumire_tip)
TIP_EVENIMENT(denumire_tip#, descriere_tip)

Ulterior, am adaugat un id fiecărui tip de eveniment (cheie primară în tabelul TIP_EVENIEMNT), iar cele două proiecții au devenit:

EVENIMENT(id_eveniment#, data_eveniment, id_tip_eveniment)
TIP_EVENIMENT(id_tip_eveniment#, denumire_tip, descriere_tip)

După realizarea normalizării în FN3 :

EVENIMENT		
id_eveniment#	data_eveniment	id_tip_eveniment
110	13-07-2021	121
111	25-06-2021	124
112	15-09-2020	125
113	03-04-2021	124

TIP_EVENIMENT		
id_tip_eveniment#	denumire_tip	descriere_tip
110	nunta	maxim 150 pers
111	revedere	liceu/facultate
112	cununie	null
113	revedere	liceu/facultate

Astfel, fiecare atribut care nu este cheie depinde direct de cheia primară, iar, acum, relația se află în FN3.

10. Crearea tabelelor în SQL și inserarea datelor

+

13. Crearea unei secvențe

--crearea tabelului clienti

```
CREATE TABLE clienti
  (id_client number(5),
  nume varchar2(25) constraint c_nume not null,
  prenume varchar2(25),
  nr_telefon char(10),
  CONSTRAINT client_pk PRIMARY KEY (id_client),
  unique(nr_telefon)
);
```

--secventa pentru generarea codurilor clientilor

```
CREATE SEQUENCE SEQ_CLIENT
INCREMENT by 1
START WITH 1
MINVALUE 1
MAXVALUE 1000
NOCYCLE;
```

--inserarea datelor in tabelul clienti

```
INSERT INTO clienti(id_client, nume, prenume, nr_telefon)
  VALUES (SEQ_CLIENT.NEXTVAL, 'Popescu', 'Maria', '0371692722');
INSERT INTO clienti(id_client, nume, prenume, nr_telefon)
  VALUES (SEQ_CLIENT.NEXTVAL, 'Radu', 'Andrei', '0772262672');
INSERT INTO clienti(id_client, nume, prenume, nr_telefon)
  VALUES (SEQ_CLIENT.NEXTVAL, 'Avram', 'Ana-Maria', '0727635283');
INSERT INTO clienti(id_client, nume, prenume, nr_telefon)
  VALUES (SEQ_CLIENT.NEXTVAL, 'Gavrila', 'Cristina', '0742830745');
INSERT INTO clienti(id_client, nume, prenume, nr_telefon)
  VALUES (SEQ_CLIENT.NEXTVAL, 'Istrate', 'Bogdan-Ionut', '0739428220');
INSERT INTO clienti(id_client, nume, prenume, nr_telefon)
  VALUES (SEQ_CLIENT.NEXTVAL, 'Comnoiu', 'Adela-Loana', '0762863289');
INSERT INTO clienti(id_client, nume, prenume, nr_telefon)
  VALUES (SEQ_CLIENT.NEXTVAL, 'Marin', 'Liviu', '0758263927');

SELECT * FROM clienti;
```

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the database structure under 'Connections' for 'ProjectBD'. The main area has two tabs: 'Worksheet' and 'Query Builder'. The 'Worksheet' tab contains the following SQL code:

```
--inserarea datelor in tabelul clienti
INSERT INTO clienti(id_client, nume, prenume, nr_telefon)
VALUES (SEQ_CLIENT.NEXTVAL, 'Popescu', 'Maria', '0371692722');
INSERT INTO clienti(id_client, nume, prenume, nr_telefon)
VALUES (SEQ_CLIENT.NEXTVAL, 'Radu', 'Andrei', '0772262672');
INSERT INTO clienti(id_client, nume, prenume, nr_telefon)
VALUES (SEQ_CLIENT.NEXTVAL, 'Avram', 'Ana-Maria', '0727635283');
INSERT INTO clienti(id_client, nume, prenume, nr_telefon)
VALUES (SEQ_CLIENT.NEXTVAL, 'Gavrilă', 'Cristina', '0742830745');
INSERT INTO clienti(id_client, nume, prenume, nr_telefon)
VALUES (SEQ_CLIENT.NEXTVAL, 'Istrate', 'Bogdan-Ionut', '0739428220');
INSERT INTO clienti(id_client, nume, prenume, nr_telefon)
VALUES (SEQ_CLIENT.NEXTVAL, 'Comniciu', 'Adela-Ioana', '0762863289');
INSERT INTO clienti(id_client, nume, prenume, nr_telefon)
VALUES (SEQ_CLIENT.NEXTVAL, 'Marin', 'Liviu', '0758263927');

SELECT * FROM clienti;

--crearea tabelului tipuri_evenimente
*CREATE TABLE tipuri_evenimente
(id_tip_eveniment number(5),
denumire_tip varchar2(20),
```

The 'Query Result' window below shows the data inserted into the 'clienti' table:

ID_CLIENT	NUME	NR_TELEFON
1	Popescu Maria	0371692722
2	Radu Andrei	0772262672
3	Avram Ana-Maria	0727635283
4	Gavrilă Cristina	0742830745
5	Istrate Bogdan-Ionut	0739428220
6	Comniciu Adela-Ioana	0762863289
7	Marin Liviu	0758263927

--crearea tabelului tipuri_evenimente

```
CREATE TABLE tipuri_evenimente
(id_tip_eveniment number(5),
denumire_tip varchar2(20),
descriere long,
CONSTRAINT tip_eveniment_pk PRIMARY KEY (id_tip_eveniment)
);
```

--inserarea datelor in tabelul tipuri_evenimente

```
INSERT INTO tipuri_evenimente
VALUES (121, 'nunta', null);
INSERT INTO tipuri_evenimente
VALUES (122, 'botez', null);
INSERT INTO tipuri_evenimente
VALUES (123, 'aniversare', 'inclusiv majorat');
INSERT INTO tipuri_evenimente
VALUES (124, 'revedere', 'liceu/facultate');
INSERT INTO tipuri_evenimente
VALUES (125, 'cununie', null);
```

SELECT * FROM tipuri_evenimente;

```

File Edit View Navigate Run Source Team Tools Window Help
Connections ProjectBD.sql ProjectBD.log Welcome Page ProjectBD
SQL Worksheet History
Worksheet - Query Builder
);
--inserarea datelor in tabelul tipuri_evenimente
INSERT INTO tipuri_evenimente
VALUES (121, 'nunta', null);
INSERT INTO tipuri_evenimente
VALUES (122, 'botez', null);
INSERT INTO tipuri_evenimente
VALUES (123, 'aniversare', 'inclusiv majorat');
INSERT INTO tipuri_evenimente
VALUES (124, 'revedere', 'liceu/facultate');
INSERT INTO tipuri_evenimente
VALUES (125, 'cununie', null);

SELECT * FROM tipuri_evenimente;

--crearea tabelului evenimente
CREATE TABLE evenimente
(id_eveniment number(5),
data_eveniment date,
preferinte long,
id_tip_eveniment number(5) CONSTRAINT c_tip_eveniment not null,
CONSTRAINT eveniment_pk PRIMARY KEY (id_eveniment),
unique(data_eveniment),
CONSTRAINT eveniment_fk foreign key(id_tip_eveniment) references
tipuri_evenimente(id_tip_eveniment)
);

```

Query Result

ID_TIP_EVENIMENT	DENUMIRE_TIP	DESCRIBE
1	121nunta	(null)
2	122botez	(null)
3	123aniversare	inclusiv majorat
4	124revedere	liceu/facultate
5	125cununie	(null)

--crearea tabelului evenimente

```

CREATE TABLE evenimente
(id_eveniment number(5),
data_eveniment date,
preferinte long,
id_tip_eveniment number(5) CONSTRAINT c_tip_eveniment not null,
CONSTRAINT eveniment_pk PRIMARY KEY (id_eveniment),
unique(data_eveniment),
CONSTRAINT eveniment_fk foreign key(id_tip_eveniment) references
tipuri_evenimente(id_tip_eveniment)
);

```

--inserarea datelor in tabelul evenimente

```

INSERT INTO evenimente
VALUES (110, to_date('13-07-2021','dd-mm-yyyy'), 'decor albastru', 122);
INSERT INTO evenimente
VALUES (111, to_date('25-06-2021','dd-mm-yyyy'), null, 125);
INSERT INTO evenimente
VALUES (112, to_date('15-09-2020','dd-mm-yyyy'), null, 124);
INSERT INTO evenimente
VALUES (113, to_date('03-04-2021','dd-mm-yyyy'), 'flori naturale', 121);
INSERT INTO evenimente
VALUES (114, to_date('28-02-2021','dd-mm-yyyy'), 'decor rosu', 123);

```

SELECT * FROM evenimente;

```

File Edit View Navigate Run Source Team Tools Window Help
Connections ProjectBD.sql ProjectBD.log Welcome Page ProjectBD
SQL Worksheet History
Worksheet Query Builder
CONSTRRAINT eveniment_fk foreign key(id_tip_eveniment) references tipuri_evenimente(id_tip_eveniment)
);

--inserarea datelor in tabelul evenimente
INSERT INTO evenimente
VALUES (110, to_date('13-07-2021','dd-mm-yyyy'), 'decor albastru', 122);
INSERT INTO evenimente
VALUES (111, to_date('25-06-2021','dd-mm-yyyy'), null, 125);
INSERT INTO evenimente
VALUES (112, to_date('15-09-2020','dd-mm-yyyy'), null, 124);
INSERT INTO evenimente
VALUES (113, to_date('03-04-2021','dd-mm-yyyy'), 'flori naturale', 121);
INSERT INTO evenimente
VALUES (114, to_date('28-02-2021','dd-mm-yyyy'), 'decor rosu', 123);

SELECT * FROM evenimente;

--crearea tabelului moduri_de_plata
CREATE TABLE moduri_de_plata
(id_mod_plata number(5),
denumire varchar2(20),
observatii long,
CONSTRAINT mod_plata_pk PRIMARY KEY (id_mod_plata),
unique(denumire)
);

Query Result
All Rows Fetched: 5 in 0.985 seconds
ID_EVENIMENT | DATA_EVENIMENT | TIP_EVENIMENT | ID_TIP_EVENIMENT
1           | 11013-JUL-21 | decor albastru | 122
2           | 11125-JUN-21 | (null)          | 125
3           | 11215-SEP-20 | (null)          | 124
4           | 11303-APR-21 | flori naturale | 121
5           | 11428-FEB-21 | decor rosu    | 123

```

--crearea tabelului moduri_de_plata

```

CREATE TABLE moduri_de_plata
(id_mod_plata number(5),
denumire varchar2(20),
observatii long,
CONSTRAINT mod_plata_pk PRIMARY KEY (id_mod_plata),
unique(denumire)
);
```

--inserarea datelor in tabelul moduri_de_plata

```

INSERT INTO moduri_de_plata
VALUES (600, 'cash', null);
INSERT INTO moduri_de_plata
VALUES (610, 'card', null);
INSERT INTO moduri_de_plata
VALUES (620, 'online', null);
INSERT INTO moduri_de_plata
VALUES (630, 'in rate', 'numai pentru evenimente');
INSERT INTO moduri_de_plata
VALUES (640, 'tichet', null);
```

```
SELECT * FROM moduri_de_plata;
```

```

File Edit View Navigate Run Source Team Tools Window Help
Connections ProjectBD.sql ProjectBD.log Welcome Page ProjectBD
SQL Worksheet History
Worksheet Query Builder
    observatii long,
    CONSTRAINT mod_plata_pk PRIMARY KEY (id_mod_plata),
    unique (denumire)
);

--inserarea datelor in tabelul moduri_de_plata
INSERT INTO moduri_de_plata
VALUES (600, 'cash', null);
INSERT INTO moduri_de_plata
VALUES (610, 'card', null);
INSERT INTO moduri_de_plata
VALUES (620, 'online', null);
INSERT INTO moduri_de_plata
VALUES (630, 'in rate', 'numai pentru evenimente');
INSERT INTO moduri_de_plata
VALUES (640, 'tichet', null);

SELECT * FROM moduri_de_plata;

--crearea tabelului comenzi
CREATE TABLE comenzi
(id_comanda number(5),

```

Query Result

ID_MOD_PLATA	DENUMIRE	OBSEVATII
1	600 cash	(null)
2	610 card	(null)
3	620 online	(null)
4	630 in rate	numai pentru evenimente
5	640 tichet	(null)

--crearea tabelului comenzi

```

CREATE TABLE comenzi
(id_comanda number(5),
data_comanda date default sysdate,
id_mod_plata number(5) constraint mod_not_null not null,
id_client number(5) constraint client_not_null not null,
id_eveniment number(5),
CONSTRAINT comenzi_pk PRIMARY KEY (id_comanda),
CONSTRAINT comanda_mod_plata_fk foreign key(id_mod_plata) references
moduri_de_plata(id_mod_plata),
CONSTRAINT comanda_client_fk foreign key(id_client) references clienti(id_client),
CONSTRAINT comanda_eveniment_fk foreign key(id_eveniment) references
evenimente(id_eveniment)
);
```

--secreta pentru generarea codurilor comenzilor

```

CREATE SEQUENCE SEQ_COMENZI
INCREMENT by 10
START WITH 10
MINVALUE 10
MAXVALUE 10000
NOCYCLE;
```

--inserarea datelor in tabelul comenzi

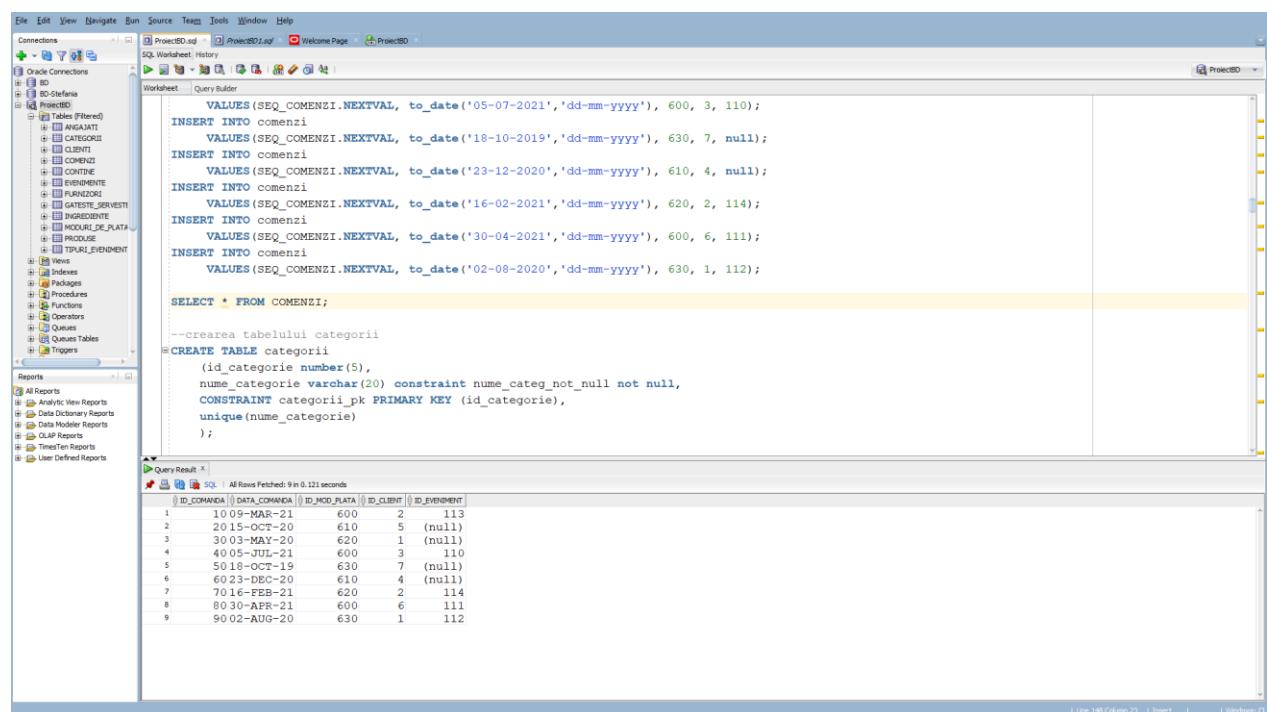
```

INSERT INTO comenzi
VALUES(SEQ_COMENZI.NEXTVAL, to_date('09-03-2021','dd-mm-yyyy'), 600, 2, 113);
INSERT INTO comenzi
VALUES(SEQ_COMENZI.NEXTVAL, to_date('15-10-2020','dd-mm-yyyy'), 610, 5, null);
```

INSERT INTO comenzi

```
VALUES(SEQ_COMENZI.NEXTVAL, to_date('03-05-2020','dd-mm-yyyy'), 620, 1, null);
INSERT INTO comenzi
VALUES(SEQ_COMENZI.NEXTVAL, to_date('05-07-2021','dd-mm-yyyy'), 600, 3, 110);
INSERT INTO comenzi
VALUES(SEQ_COMENZI.NEXTVAL, to_date('18-10-2019','dd-mm-yyyy'), 630, 7, null);
INSERT INTO comenzi
VALUES(SEQ_COMENZI.NEXTVAL, to_date('23-12-2020','dd-mm-yyyy'), 610, 4, null);
INSERT INTO comenzi
VALUES(SEQ_COMENZI.NEXTVAL, to_date('16-02-2021','dd-mm-yyyy'), 620, 2, 114);
INSERT INTO comenzi
VALUES(SEQ_COMENZI.NEXTVAL, to_date('30-04-2021','dd-mm-yyyy'), 600, 6, 111);
INSERT INTO comenzi
VALUES(SEQ_COMENZI.NEXTVAL, to_date('02-08-2020','dd-mm-yyyy'), 630, 1, 112);
```

```
SELECT * FROM COMENZI;
```



--crearea tabelului categorii

```
CREATE TABLE categorii
(id_categorie number(5),
nume_categorie varchar(20) constraint nume_categ_not_null not null,
CONSTRAINT categorii_pk PRIMARY KEY (id_categorie),
unique(nume_categorie)
);
```

--sevența pentru generarea codurilor categoriilor

CREATE SEQUENCE SEQ_CATEGORIES

INCREMENT by 10

```

START WITH 200
MINVALUE 200
MAXVALUE 10000
NOCYCLE;

```

```

--inserarea datelor in tabelul categorii
INSERT INTO categorii
    VALUES (SEQ_CATEGORII.NEXTVAL, 'paste');
INSERT INTO categorii
    VALUES (SEQ_CATEGORII.NEXTVAL, 'salate');
INSERT INTO categorii
    VALUES (SEQ_CATEGORII.NEXTVAL, 'pizza');
INSERT INTO categorii
    VALUES (SEQ_CATEGORII.NEXTVAL, 'desert');
INSERT INTO categorii
    VALUES (SEQ_CATEGORII.NEXTVAL, 'bauturi');
INSERT INTO categorii
    VALUES (SEQ_CATEGORII.NEXTVAL, 'friptura');

SELECT * FROM categorii;

```

The screenshot shows the Oracle SQL Developer interface. On the left, the Object Navigator pane displays various database objects like tables, packages, and procedures. The central workspace contains the following SQL code:

```

MAXVALUE 10000
NOCYCLE;

--inserarea datelor in tabelul categorii
INSERT INTO categorii
    VALUES (SEQ_CATEGORII.NEXTVAL, 'paste');
INSERT INTO categorii
    VALUES (SEQ_CATEGORII.NEXTVAL, 'salate');
INSERT INTO categorii
    VALUES (SEQ_CATEGORII.NEXTVAL, 'pizza');
INSERT INTO categorii
    VALUES (SEQ_CATEGORII.NEXTVAL, 'desert');
INSERT INTO categorii
    VALUES (SEQ_CATEGORII.NEXTVAL, 'bauturi');
INSERT INTO categorii
    VALUES (SEQ_CATEGORII.NEXTVAL, 'friptura');

SELECT * FROM categorii;

--crearea tabelului furnizori
CREATE TABLE furnizori
    (id_furnizor number(5),
    denumire varchar2(20) constraint denumiref_not_null not null,
    nr_telefon char(10) constraint nr_telf_not_null not null,
    unique(nr_telefon),
    unique(denumire));

```

The bottom pane shows the "Query Result" window with the following data:

ID_CATEGORII	NAME_CATEGORIE
1	240bauturi
2	230desert
3	250friptura
4	200paste
5	220pizza
6	210salate

```

--crearea tabelului furnizori
CREATE TABLE furnizori
    (id_furnizor number(5),
    denumire varchar2(20) constraint denumiref_not_null not null,
    nr_telefon char(10) constraint nr_telf_not_null not null,
    unique(nr_telefon),
    unique(denumire),

```

```

CONSTRAINT furnizori_pk PRIMARY KEY (id_furnizor)
);

--secreta pentru generarea codurilor furnizorilor
CREATE SEQUENCE SEQ_FURNIZOR
INCREMENT by 10
START WITH 1000
MINVALUE 1000
MAXVALUE 90000
NOCYCLE;

--inserarea datelor in tabelul furnizori
INSERT INTO furnizori
VALUES (SEQ_FURNIZOR.NEXTVAL, 'SC Bucuria Gustului', '0764652224');
INSERT INTO furnizori
VALUES (SEQ_FURNIZOR.NEXTVAL, 'SC Gustos din Natura', '0775375472');
INSERT INTO furnizori
VALUES (SEQ_FURNIZOR.NEXTVAL, 'SC CrisFoods', '0775264221');
INSERT INTO furnizori
VALUES (SEQ_FURNIZOR.NEXTVAL, 'SC De Bun Gust', '0787625242');
INSERT INTO furnizori
VALUES (SEQ_FURNIZOR.NEXTVAL, 'SC Delicios', '0769727366');

SELECT * FROM furnizori;

```

The screenshot shows the Oracle SQL Developer environment. In the left sidebar, under 'Connections', there is a connection named 'ProjectBD'. Under 'Schemas', there is a schema named 'PROJECTBD'. The main workspace contains the following SQL code:

```

MINVALUE 1000
MAXVALUE 90000
NOCYCLE;

--inserarea datelor in tabelul furnizori
INSERT INTO furnizori
VALUES (SEQ_FURNIZOR.NEXTVAL, 'SC Bucuria Gustului', '0764652224');
INSERT INTO furnizori
VALUES (SEQ_FURNIZOR.NEXTVAL, 'SC Gustos din Natura', '0775375472');
INSERT INTO furnizori
VALUES (SEQ_FURNIZOR.NEXTVAL, 'SC CrisFoods', '0775264221');
INSERT INTO furnizori
VALUES (SEQ_FURNIZOR.NEXTVAL, 'SC De Bun Gust', '0787625242');
INSERT INTO furnizori
VALUES (SEQ_FURNIZOR.NEXTVAL, 'SC Delicios', '0769727366');

SELECT * FROM furnizori;

```

Below the code, the 'Reports' section is visible, showing various report types like All Reports, Data Dictionary Reports, OLAP Reports, and Timetrend Reports.

In the bottom right corner of the workspace, there is a 'Query Result' tab. It displays the following table:

ID_FURNIZOR	DENUMIRE	NR_TELEFON
1	1000 SC Bucuria Gustului	0764652224
2	1010 SC Gustos din Natura	0775375472
3	1020 SC CrisFoods	0775264221
4	1030 SC De Bun Gust	0787625242
5	1040 SC Delicios	0769727366

--crearea tabelului ingrediente

```

CREATE TABLE ingrediente
(id_ingredient number(5),
denumire varchar2(20) constraint denumire_i_not_null not null,

```

```
denumire varchar2(20) constraint denumire_i_not_null not null,
id_furnizor number(5) constraint furnizor_not_null not null,
CONSTRAINT ingrediente_pk PRIMARY KEY (id_ingredient),
CONSTRAINT ingred_furnizor_fk foreign key(id_furnizor) references furnizori(id_furnizor),
unique(denumire)
);
```

--secreta pentru generarea codurilor ingredientelor

```
CREATE SEQUENCE SEQ_INGREDIENT
INCREMENT by 1
START WITH 170
MINVALUE 170
MAXVALUE 90000
NOCYCLE;
```

--inserarea datelor in tabelul ingrediente

```
INSERT INTO ingrediente
VALUES(SEQ_INGREDIENT.NEXTVAL, 'spaghete', 1000);
INSERT INTO ingrediente
VALUES(SEQ_INGREDIENT.NEXTVAL, 'sos de rosii', 1000);
INSERT INTO ingrediente
VALUES(SEQ_INGREDIENT.NEXTVAL, 'carne tocata', 1010);
INSERT INTO ingrediente
VALUES(SEQ_INGREDIENT.NEXTVAL, 'parmezan', 1020);
INSERT INTO ingrediente
VALUES(SEQ_INGREDIENT.NEXTVAL, 'salata verde', 1030);
INSERT INTO ingrediente
VALUES(SEQ_INGREDIENT.NEXTVAL, 'piept de pui', 1020);
INSERT INTO ingrediente
VALUES(SEQ_INGREDIENT.NEXTVAL, 'rosii', 1000);
INSERT INTO ingrediente
VALUES(SEQ_INGREDIENT.NEXTVAL, 'castraveti', 1040);
INSERT INTO ingrediente
VALUES(SEQ_INGREDIENT.NEXTVAL, 'branza', 1040);
INSERT INTO ingrediente
VALUES(SEQ_INGREDIENT.NEXTVAL, 'faina', 1020);
INSERT INTO ingrediente
VALUES(SEQ_INGREDIENT.NEXTVAL, 'masline', 1030);
INSERT INTO ingrediente
VALUES(SEQ_INGREDIENT.NEXTVAL, 'piscoturi', 1000);
INSERT INTO ingrediente
VALUES(SEQ_INGREDIENT.NEXTVAL, 'mascarpone', 1010);
INSERT INTO ingrediente
VALUES(SEQ_INGREDIENT.NEXTVAL, 'ciocolata', 1010);
INSERT INTO ingrediente
VALUES(SEQ_INGREDIENT.NEXTVAL, 'apa', 1030);
INSERT INTO ingrediente
```

```

VALUES(SEQ_INGREDIENT.NEXTVAL, 'suc', 1040);
INSERT INTO ingrediente
VALUES(SEQ_INGREDIENT.NEXTVAL, 'vin', 1010);
INSERT INTO ingrediente
VALUES(SEQ_INGREDIENT.NEXTVAL, 'carne porc', 1020);
INSERT INTO ingrediente
VALUES(SEQ_INGREDIENT.NEXTVAL, 'carne pui', 1020);

SELECT * FROM ingrediente;

```

The screenshot shows the Oracle SQL Developer interface. The left sidebar displays the database schema with various tables like 'Tabels (Ferea)', 'CATEGORII', 'CLIENTI', 'COMENZI', 'CONTINE', 'EVENTIMENTE', 'FURNIZORI', 'GATUTE_SERVESTI', 'LOCATII', 'MODERATORI', 'PRODUSE', 'STOCARE', 'TIPURI_EVENTIMENT', 'VIEV', 'VIEWS', 'VIEWS', 'PACKAGES', 'PROCEDURES', 'FUNCTIONS', 'OPERATORS', 'TRIGGERS', 'QUEUES_TABLES', and 'TRIGGERS'. The central workspace contains the following SQL code:

```

INSERT INTO ingrediente
VALUES(SEQ_INGREDIENT.NEXTVAL, 'ciocolata', 1010);
INSERT INTO ingrediente
VALUES(SEQ_INGREDIENT.NEXTVAL, 'apa', 1030);
INSERT INTO ingrediente
VALUES(SEQ_INGREDIENT.NEXTVAL, 'suc', 1040);
INSERT INTO ingrediente
VALUES(SEQ_INGREDIENT.NEXTVAL, 'vin', 1010);
INSERT INTO ingrediente
VALUES(SEQ_INGREDIENT.NEXTVAL, 'carne porc', 1020);
INSERT INTO ingrediente
VALUES(SEQ_INGREDIENT.NEXTVAL, 'carne pui', 1020);

SELECT * FROM ingrediente;

--crearea tabelului produse
CREATE TABLE produse
(id_produs number(5),

```

The 'Query Result' tab shows the output of the SELECT query:

ID_INGREDIENT	DENUMIRE	ID_CATEGORIE
1	170 spaghetti	1000
2	171 sos de rosii	1000
3	172 carne tocata	1010
4	173 parmezan	1020
5	174 salata verde	1030
6	175 pierte de pui	1020
7	176 rosii	1000
8	177 castraveti	1040
9	178 branza	1040
10	179 faina	1020
11	180 masline	1030
12	181 picoturi	1000
13	182 cascavoline	1010
14	183 ciocolata	1010
15	184 apa	1030
16	185 suc	1040
17	186 vin	1010
18	187 carne porc	1020
19	188 carne pui	1020

--crearea tabelului produse

```

CREATE TABLE produse
(id_produs number(5),
denumire varchar2(30) constraint denumire_p_not_null not null,
pret number(3) constraint pret_not_null not null,
observatii long,
id_categorie number(5) constraint categ_not_null not null,
unique(denumire),
check(pret>0),
CONSTRAINT produse_pk PRIMARY KEY (id_produs),
CONSTRAINT produs_categ_fk foreign key(id_categorie) references
categorie(id_categorie)
);

```

--secreta pentru generarea codurilor produselor

```

CREATE SEQUENCE SEQ_PRODUS

```

```

INCREMENT by 1

```

```

START WITH 50

```

```

MINVALUE 50
MAXVALUE 90000
NOCYCLE;

--inserarea datelor in tabelul produse
INSERT INTO produse
VALUES(SEQ_PRODUS.NEXTVAL, 'paste carbonara', 24, '300g', 200);
INSERT INTO produse
VALUES(SEQ_PRODUS.NEXTVAL, 'paste bolognese', 22, null, 200);
INSERT INTO produse
VALUES(SEQ_PRODUS.NEXTVAL, 'salata cu pui', 23, '200g', 210);
INSERT INTO produse
VALUES(SEQ_PRODUS.NEXTVAL, 'salata greceasca', 16, '200g', 210);
INSERT INTO produse
VALUES(SEQ_PRODUS.NEXTVAL, 'pizza margherita', 17, 'medie', 220);
INSERT INTO produse
VALUES(SEQ_PRODUS.NEXTVAL, 'pizza capriciosa', 19, 'medie', 220);
INSERT INTO produse
VALUES(SEQ_PRODUS.NEXTVAL, 'tiramisu', 15, null, 230);
INSERT INTO produse
VALUES(SEQ_PRODUS.NEXTVAL, 'clatite', 14, null, 230);
INSERT INTO produse
VALUES(SEQ_PRODUS.NEXTVAL, 'apa', 5, '500ml', 240);
INSERT INTO produse
VALUES(SEQ_PRODUS.NEXTVAL, 'suc', 7, 'diverse sortimente', 240);
INSERT INTO produse
VALUES(SEQ_PRODUS.NEXTVAL, 'vin', 10, 'la sticla', 240);
INSERT INTO produse
VALUES(SEQ_PRODUS.NEXTVAL, 'friptura pui', 25, '230g', 250);
INSERT INTO produse
VALUES(SEQ_PRODUS.NEXTVAL, 'friptura porc', 28, '200g', 250);

SELECT * FROM produse;

```

```

File Edit View Navigate Run Source Team Tools Window Help
Connections ProjectID.scd ProjectID.sqf Welcome Page ProjectID
SQL Worksheet History
Worksheet Query Builder
INSERT INTO produse
VALUES (SEQ_PRODUS.NEXTVAL, 'pizza capriciosa', 19, 'medie', 220);
INSERT INTO produse
VALUES (SEQ_PRODUS.NEXTVAL, 'tiramisu', 15, null, 230);
INSERT INTO produse
VALUES (SEQ_PRODUS.NEXTVAL, 'clatite', 14, null, 230);
INSERT INTO produse
VALUES (SEQ_PRODUS.NEXTVAL, 'apa', 5, '500ml', 240);
INSERT INTO produse
VALUES (SEQ_PRODUS.NEXTVAL, 'suc', 7, 'diverse sortimente', 240);
INSERT INTO produse
VALUES (SEQ_PRODUS.NEXTVAL, 'vin', 10, 'la sticla', 240);
INSERT INTO produse
VALUES (SEQ_PRODUS.NEXTVAL, 'riptura pui', 25, '230g', 250);
INSERT INTO produse
VALUES (SEQ_PRODUS.NEXTVAL, 'riptura porc', 28, '200g', 250);

SELECT * FROM produse;

--crearea tabelului contine
CREATE TABLE contine

```

Query Result:

ID_PRODUS	DENUMIRE	PRET	OBSERVATII	ID_CATEGORIE
1	50paste carbonara	24	300g	200
2	51paste bolognese	22	(null)	200
3	52salata cu pui	23	200g	210
4	53salata greceasca	16	200g	210
5	54pizza margherita	17	200g	220
6	55pizza capriciosa	19	medie	220
7	56tiramisu	15	(null)	230
8	57clatite	14	(null)	230
9	58apa	5	500ml	240
10	59suc	7	diverse sortimente	240
11	60vin	10	la sticla	240
12	61riptura pui	25	230g	250
13	62riptura porc	28	200g	250

--crearea tabelului contine

CREATE TABLE contine

```

(id_produs number(5) constraint produs_not_null not null,
id_ingredient number(5) constraint ingred_not_null not null,
CONSTRAINT contine_ingred_fk foreign key(id_ingredient) references
ingrediente(id_ingredient),
CONSTRAINT contine_produs_fk foreign key(id_produs) references produse(id_produs),
CONSTRAINT pk_contine primary key(id_produs, id_ingredient)
);

```

--inserarea datelor in tabelul contine

INSERT INTO contine

VALUES(50,170);

INSERT INTO contine

VALUES(50,173);

INSERT INTO contine

VALUES(51,170);

INSERT INTO contine

VALUES(51,171);

INSERT INTO contine

VALUES(51,172);

INSERT INTO contine

VALUES(52,174);

INSERT INTO contine

VALUES(52,175);

INSERT INTO contine

VALUES(53,174);

INSERT INTO contine

```
VALUES(53,176);
INSERT INTO contine
    VALUES(53,177);
INSERT INTO contine
    VALUES(53,178);
INSERT INTO contine
    VALUES(54,179);
INSERT INTO contine
    VALUES(54,171);
INSERT INTO contine
    VALUES(54,173);
INSERT INTO contine
    VALUES(55,179);
INSERT INTO contine
    VALUES(55,180);
INSERT INTO contine
    VALUES(55,171);
INSERT INTO contine
    VALUES(55,173);
INSERT INTO contine
    VALUES(56,181);
INSERT INTO contine
    VALUES(56,182);
INSERT INTO contine
    VALUES(57,179);
INSERT INTO contine
    VALUES(57,183);
INSERT INTO contine
    VALUES(58,184);
INSERT INTO contine
    VALUES(59,185);
INSERT INTO contine
    VALUES(60,186);

SELECT * FROM contine;
```

```

File Edit View Navigate Run Source Team Tools Window Help
Connections ProjectBD.sql ProjectBD.log Welcome Page ProjectBD
SQL Worksheet History
ProjectBD
Worksheet Query Builder
VALUES (57,183);
INSERT INTO contine
VALUES (58,184);
INSERT INTO contine
VALUES (59,185);
INSERT INTO contine
VALUES (60,186);

SELECT * FROM contine;

--crearea tabelului angajati
CREATE TABLE angajati
(id_angajat number(5),
nume varchar(25) constraint nume_ang_not_null not null,
prenume varchar(25),
email varchar(50),
nr_telefon char(10) constraint nr_ang_not_null not null,
data_angajare date default sysdate,
tip_angajat varchar2(10) constraint tip_not_null not null,
specializare varchar2(50),
experienta number(2),
CONSTRAINT angajati_pk PRIMARY KEY (id_angajat),
unique(email),
unique(nr_telefon)
);

Query Result X
SQL All Rows Fetched: 25 in 0.121 seconds
ID_FOCUS ID_INGREDIENT
1 50 170
2 50 173
3 51 170
4 51 171
5 51 172
6 52 174
7 52 175
8 53 174
9 53 176
10 53 177
11 53 178
12 54 171
13 54 173
14 54 179
15 55 171
16 55 173
17 55 179
18 55 180
19 56 181
20 56 182
21 57 179
22 57 183
23 58 184
24 59 185
25 60 186

```

--crearea tabelului angajati

```

CREATE TABLE angajati
(id_angajat number(5),
nume varchar(25) constraint nume_ang_not_null not null,
prenume varchar(25),
email varchar(50),
nr_telefon char(10) constraint nr_ang_not_null not null,
data_angajare date default sysdate,
tip_angajat varchar2(10) constraint tip_not_null not null,
specializare varchar2(50),
experienta number(2),
CONSTRAINT angajati_pk PRIMARY KEY (id_angajat),
unique(email),
unique(nr_telefon)
);
```

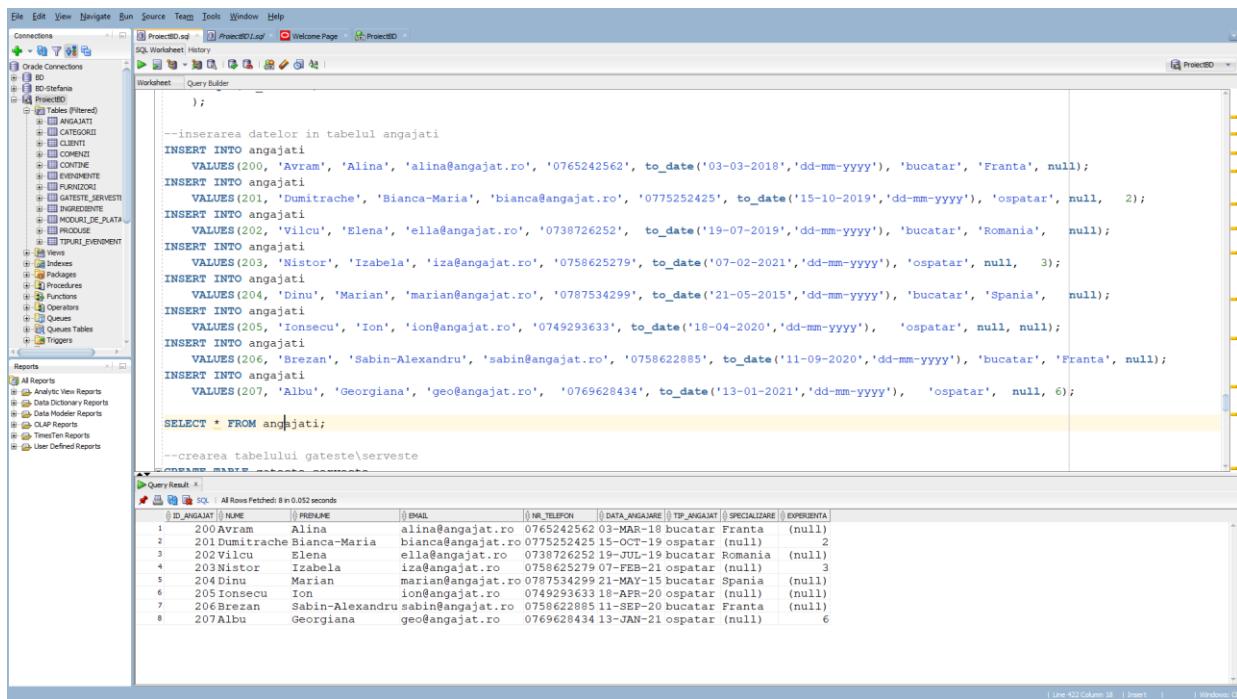
--inserarea datelor in tabelul angajati

```

INSERT INTO angajati
VALUES(200,      'Avram', 'Alina', 'alina@angajat.ro', '0765242562', to_date('03-03-2018','dd-mm-yyyy'), 'bucatar', 'Franta', null);
INSERT INTO angajati
VALUES(201,      'Dumitache', 'Bianca-Maria', 'bianca@angajat.ro', '0775252425', to_date('15-10-2019','dd-mm-yyyy'), 'ospatar', null,      2);
INSERT INTO angajati
VALUES(202,      'Vilcu', 'Elena', 'ella@angajat.ro', '0738726252', to_date('19-07-2019','dd-mm-yyyy'), 'bucatar', 'Romania', null);
INSERT INTO angajati
```

```
VALUES(203,      'Nistor', 'Izabela', 'iza@angajat.ro', '0758625279', to_date('07-02-2021','dd-mm-yyyy'), 'ospatar', null, 3);
INSERT INTO angajati
    VALUES(204,      'Dinu', 'Marian', 'marian@angajat.ro', '0787534299', to_date('21-05-2015','dd-mm-yyyy'), 'bucatar', 'Spania',    null);
INSERT INTO angajati
    VALUES(205,      'Ionsecu', 'Ion', 'ion@angajat.ro',     '0749293633', to_date('18-04-2020','dd-mm-yyyy'), 'ospatar', null, null);
INSERT INTO angajati
    VALUES(206,      'Brezan', 'Sabin-Alexandru', 'sabin@angajat.ro', '0758622885', to_date('11-09-2020','dd-mm-yyyy'), 'bucatar', 'Franta', null);
INSERT INTO angajati
    VALUES(207,      'Albu', 'Georgiana', 'geo@angajat.ro',       '0769628434', to_date('13-01-2021','dd-mm-yyyy'), 'ospatar',    null, 6);
```

```
SELECT * FROM angajati;
```



--crearea tabelului gateste\serveste

CREATE TABLE gateste serveste

```
(id_angajat number(5) constraint ang_g_not_null not null,  
id_comanda number(5) constraint comanda_g_not_null not null,  
id_produs number(5) constraint produs_g_not_null not null,  
cantitate number(2) constraint c_cantitate not null,  
CONSTRAINT gateste_angajat_fk foreign key(id_angajat) references angajati(id_angajat),  
CONSTRAINT gateste_produs_fk foreign key(id_produs) references produse(id_produs),  
CONSTRAINT gateste_comanda_fk foreign key(id_comanda) references  
comenzi(id_comanda),  
CONSTRAINT pk_gateste primary key(id_angajat, id_comanda,id_produs),
```

```
check(cantitate>0)
);

--inserarea datelor in tabelul gateste_serveste
INSERT INTO gateste_serveste
    VALUES(204,10,62,20);
INSERT INTO gateste_serveste
    VALUES(206,10,56,15);
INSERT INTO gateste_serveste
    VALUES(207,10,59,20);
INSERT INTO gateste_serveste
    VALUES(200,20,55,2);
INSERT INTO gateste_serveste
    VALUES(202,30,57,4);
INSERT INTO gateste_serveste
    VALUES(205,30,59,4);
INSERT INTO gateste_serveste
    VALUES(204,40,61,10);
INSERT INTO gateste_serveste
    VALUES(203,40,60,8);
INSERT INTO gateste_serveste
    VALUES(200,50,51,3);
INSERT INTO gateste_serveste
    VALUES(206,60,53,6);
INSERT INTO gateste_serveste
    VALUES(204,70,55,16);
INSERT INTO gateste_serveste
    VALUES(207,70,59,16);
INSERT INTO gateste_serveste
    VALUES(206,80,62,22);
INSERT INTO gateste_serveste
    VALUES(201,80,60,22);
INSERT INTO gateste_serveste
    VALUES(202,90,50,5);
INSERT INTO gateste_serveste
    VALUES(201,90,58,7);
INSERT INTO gateste_serveste
    VALUES(207,10,62,20);
INSERT INTO gateste_serveste
    VALUES(207,10,56,15);
INSERT INTO gateste_serveste
    VALUES(201,20,55,2);
INSERT INTO gateste_serveste
    VALUES(205,30,57,4);
INSERT INTO gateste_serveste
    VALUES(203,40,61,10);
INSERT INTO gateste_serveste
```

```

VALUES(201,50,51,3);
INSERT INTO gateste_serveste
    VALUES(205,60,53,6);
INSERT INTO gateste_serveste
    VALUES(207,70,55,16);
INSERT INTO gateste_serveste
    VALUES(203,80,62,22);
INSERT INTO gateste_serveste
    VALUES(205,90,50,5);
INSERT INTO gateste_serveste
    VALUES(203,10,58,3);

```

```

SELECT * FROM gateste_serveste;
COMMIT;

```

The screenshot shows the Oracle SQL Developer interface. On the left, the Object Navigator displays various database objects like tables, views, and procedures. The central area contains two tabs: 'Query Builder' and 'Query Result'. The 'Query Result' tab shows the output of the executed SQL script, which inserts data into the 'gateste_serveste' table and then selects all rows from it. The result set is a table with columns: ID_ANAGIATI, ID_COMMANDA, ID_PRODUS, and CANTITATE. The data is as follows:

ID_ANAGIATI	ID_COMMANDA	ID_PRODUS	CANTITATE
1	204	10	62
2	206	10	56
3	207	10	59
4	200	20	55
5	202	30	57
6	205	30	59
7	204	40	61
8	203	40	60
9	200	50	51
10	206	60	53
11	204	70	55
12	207	70	59
13	206	80	62
14	201	80	60
15	202	90	50
16	201	90	58
17	207	10	62
18	207	10	56
19	201	20	55
20	205	30	57
21	203	40	61
22	201	50	51
23	205	60	53
24	207	70	55
25	203	80	62
26	205	90	50
27	203	10	58

11. Cereri SQL

1. Sa se listeze toti furnizorii (id-ul si denumirea) care au furnizat ingrediente ce au fost continute de produsele din comanda clientilor cu numele "Popescu" si sunt comenzi date pentru evenimente. Rezultatul se va ordona in functie de id-ul furnizorului.

```

SELECT DISTINCT f.id_furnizor, f.denumire
FROM furnizori f JOIN ingredientete i ON (f.id_furnizor=i.id_furnizor)
JOIN contine c ON (i.id_ingredient=c.id_ingredient)
JOIN produse p ON (p.id_produs=c.id_produs)
JOIN gateste_serveste g ON (p.id_produs=g.id_produs)
JOIN comenzi co ON (g.id_comanda=co.id_comanda)
JOIN clienti cli ON (cli.id_client=co.id_client)
WHERE UPPER(cli.nume)='POPESCU' AND co.id_eveniment IS NOT NULL
ORDER BY f.id_furnizor;

```

```

File Edit View Navigate Run Source Tools Window Help
ProjectDB1.sql ProjectDB1.dwg Welcome Page ProjectDB1
Connections Oracle Connections ProjectDB1
+ ProjectDB1
  + Tables/FB
    + Views
    + Indexes
    + Packages
    + Triggers
    + Functions
    + Operators
    + Queues
    + Queues Tab
    + Types
    + Sequences
    + Materialized
    + Public Synonyms
    + Database Links
    + Public Data
    + Directories
    + Application
    + Java
Reports All Reports
  + Analytic View Reports
  + Data Dictionary Reports
  + Data Manipulator Reports
  + OLAP Reports
  + TimetTen Reports
  + User Defined Reports
Worksheet Query Builder
1 --EX 11: Cereri SQL
2 --1. Sa se listeze toti furnizorii (id-ul si denumirea) care au furnizat ingrediente ce au fost continute de produsele din comanda
3 --clientilor cu numele "Popescu" si sunt comenzi date pentru evenimente. Rezultatul se va ordona in functie de id-ul furnizorului.
4
5 SELECT DISTINCT f.id_furnizor, f.denumire
6   FROM furnizori f JOIN ingrediente i ON (f.id_furnizor=i.id_furnizor)
7     JOIN contine c ON (i.id_ingredient=c.id_ingredient)
8       JOIN produse p ON (p.id_produs=c.id_produs)
9         JOIN gateste_serveste g ON (p.id_produs=g.id_produs)
10        JOIN comenzi co ON (g.id_comanda=co.id_comanda)
11          JOIN clienti cli ON (cli.id_client=co.id_client)
12            WHERE UPPER(cli.nume)='POPESCU' AND co.id_eveniment IS NOT NULL
13 ORDER BY f.id_furnizor;
14
15
16 --2.Pentru fiecare comanda care are ziua plasarii mai mare sau egala cu 15 sa se afiseze numarul de bucatari si numarul de ospatari care au gatit/servit produsele comandate.
17 --care au gatit/servit produsele comandate.
18
19 SELECT c.id_comanda, (SELECT COUNT( DISTINCT g.id_angajat)
20   FROM angajati a JOIN gateste_serveste g ON (a.id_angajat=g.id_angajat)
21     WHERE tip_angajat='bucatar' and c.id_comanda=g.id_comanda

```

All Rows Fetched: 3 in 0.37 seconds

ID_FURNIZOR	DENUMIRE
1	1000 SC Bucuria Gustului
2	1020 SC CrisFoods
3	1030 SC De Bun Gust

2.Pentru fiecare comanda care are ziua plasarii mai mare sau egala cu 15 sa se afiseze numarul de bucatari si numarul de ospatari care au gatit/servit produsele comandate. Rezultatul se va ordona in functie de id-ul comenzi.

```

SELECT c.id_comanda, (SELECT COUNT( DISTINCT g.id_angajat)
  FROM angajati a JOIN gateste_serveste g ON (a.id_angajat=g.id_angajat)
    WHERE tip_angajat='bucatar' and c.id_comanda=g.id_comanda
  ) Nr_bucatari,
  (SELECT COUNT( DISTINCT g.id_angajat)
  FROM angajati a JOIN gateste_serveste g ON (a.id_angajat=g.id_angajat)
    WHERE tip_angajat='ospatar' and c.id_comanda=g.id_comanda
  ) Nr_ospatari
FROM comenzi c
WHERE TO_NUMBER(SUBSTR(TO_CHAR(c.data_comanda), 1, 2)) >=15
ORDER BY c.id_comanda;

```

```

File Edit View Navigate Run Source Team Tools Window Help
Connections ProjectBD.sql ProjectBD.log Welcome Page ProjectBD
Oracle Connectors
ProjectBD
Schema
Tables
Views
Indexes
Packages
Procedures
Functions
Triggers
Queues
Types
Synonyms
Materialized
Materialized
Synonyms
Public Data
Delete Log
Public Data
Directories
Editions
Application
Java
Worksheet Query Builder
14
15
16 --2. Pentru fiecare comanda care are ziua plasarii mai mare sau egala cu 15 sa se afiseze numarul de bucatari si numarul de ospatari
17 --care au gatit/servit produsele comandate. Rezultatul se va ordona in functie de id-ul comenzii.
18
19 SELECT c.id_comanda, (SELECT COUNT(DISTINCT g.id_angajat)
20   FROM angajati a JOIN gateste_serveste g ON (a.id_angajat=g.id_angajat)
21   WHERE tip_angajat='bucatar' AND c.id_comanda=g.id_comanda
22 ) Nr_bucatari,
23 (SELECT COUNT(DISTINCT g.id_angajat)
24   FROM angajati a JOIN gateste_serveste g ON (a.id_angajat=g.id_angajat)
25   WHERE tip_angajat='ospatar' AND c.id_comanda=g.id_comanda
26 ) Nr_ospatari
27 FROM comenzi c
28 WHERE TO_NUMBER(SUBSTR(TO_CHAR(c.data_comanda), 1, 2)) >=15
29 ORDER BY c.id_comanda;
30
31
32 --3. Sa se afiseze cate produse din fiecare categorie au fost comandate pentru evenimente de tip nunta care au avut loc in anul 2021.
33 --Rezultatul va fi de forma "Din categoria *nume_categorie* au fost comandate *nr* produse."
34

```

All Reports
Analytic View Report
Data Dictionary Report
Data Modeler Report
OLAP Reports
Trend Report
User Defined Report

Query Results X

ID_COMANDA	NR_BUCATARIS	NR_OSPATARI
1	20	1
2	50	1
3	60	1
4	70	1
5	80	2

Saved: C:\Users\stefan\ProjectBD1.sql

3. Sa se afiseze cate produse din fiecare categorie au fost comandate pentru evenimente de tip nunta care au avut loc in anul 2021. Rezultatul va fi de forma "Din categoria *nume_categorie* au fost comandate *nr* produse."

```

SELECT 'Din categoria ' || nume_categorie || ' au fost comandate ' ||
TO_CHAR(SUM(cantitate)) || ' produse.'
FROM categorii c JOIN produse p ON (p.id_categoria=c.id_categoria)
JOIN gateste_serveste g ON (g.id_produs=p.id_produs)
JOIN angajati a ON (g.id_angajat=a.id_angajat)
WHERE a.tip_angajat='ospatar' AND id_comanda IN (SELECT id_comanda
FROM comenzi co JOIN evenimente e ON
(e.id_eveniment=co.id_eveniment)
JOIN tipuri_evenimente t ON
(e.id_tip_eveniment=t.id_tip_eveniment)
WHERE t.denumire_tip='nunta' AND
SUBSTR(TO_CHAR(data_eveniment),8,9)='21'
)
GROUP BY nume_categorie;

```

```

File Edit View Navigate Run Source Team Tools Window Help
Connections ProjectDB.sql ProjectDB Log Welcome Page ProjectDB
+ Oracle Connections
  + ID-Deferrals
    + ProjectDB
      + Database Properties
      + Views
      + Indexes
      + Packages
      + Procedures
      + Functions
      + Operators
      + Queues
      + Types
      + Sequences
      + Materialized Views
      + Materialized Views Log
      + Public Synonyms
      + Database Links
      + Directories
      + Triggers
      + Java
Reports All Reports
  + Application View Reports
  + Data Comparison Reports
  + Data Modeler Reports
  + OLAP Reports
  + TimeTen Reports
  + User Defined Reports
SQL Worksheet History
Worksheet Query Builder
29 ORDER BY c.id_comanda;
30
31
32 ---3. Sa se afiseze cate produse din fiecare categorie au fost comandate pentru evenimente de tip nunta care au avut loc in anul 2021.
33 --Rezultatul va fi de forma "Din categoria *nume_categorie* au fost comandate *nr* produse."
34
35 SELECT 'Din categoria ' || nume_categorie || ' au fost comandate ' || TO_CHAR(SUM(cantitate)) || ' produse.'
  FROM categorii c JOIN produse p ON (p.id_categorie=c.id_categorie)
  JOIN gateste_serveste g ON (g.id_produs=p.id_produs)
  JOIN angajati a ON (g.id_angajat=a.id_angajat)
  WHERE a.tip_angajat='ospatar' AND id_comanda IN (SELECT id_comanda
    FROM comenzi c JOIN evenimente e ON (e.id_eveniment=c.id_eveniment)
    JOIN tipuri_evenimente t ON (e.id_tip_eveniment=t.id_tip_eveniment)
    WHERE t.denumire_tip='nunta' AND SUBSTR(TO_CHAR(data_eveniment),8,9)='21'
  )
  GROUP BY nume_categorie;
36
37
38
39
40
41
42
43
44
45
46
47 ---4. Pentru fiecare an din perioada 2019-2021 sa se afiseze totalul incasarilor restaurantului.
48 --varianta1, folosind having
49 WITH incasari2019 AS (SELECT TO_CHAR(data_comanda,'YYYY'), SUM(cantitate*pret) Suma

```

All Rows Fetched: 3 in 0.304 seconds

DIN CATEGORIA	[NUME_CATEGORIE]	[AU FOST COMANDATE]	[TO_CHAR(SUM(CANTITATE))]	[PRODUSE]
1	Din categoria bauturi	au fost comandate	23	produse.
2	Din categoria desert	au fost comandate	15	produse.
3	Din categoria fructura	au fost comandate	20	produse.

4. Pentru fiecare an din perioada 2019-2021 sa se afiseze totalul incasarilor restaurantului.

--varianta1, folosind having

```

WITH incasari2019 AS (SELECT TO_CHAR(data_comanda,'YYYY'), SUM(cantitate*pret) Suma
  FROM gateste_serveste g JOIN comenzi c ON (g.id_comanda=c.id_comanda)
  JOIN produse p ON (p.id_produs=g.id_produs)
  JOIN angajati a ON (g.id_angajat=a.id_angajat)
  WHERE tip_angajat='ospatar'
  GROUP BY TO_CHAR(data_comanda,'YYYY')
  HAVING TO_CHAR(data_comanda,'YYYY')=2019
),

```

```

incasari2020 AS (SELECT TO_CHAR(data_comanda,'YYYY'), SUM(cantitate*pret) Suma
  FROM gateste_serveste g JOIN comenzi c ON (g.id_comanda=c.id_comanda)
  JOIN produse p ON (p.id_produs=g.id_produs)
  JOIN angajati a ON (g.id_angajat=a.id_angajat)
  WHERE tip_angajat='ospatar'
  GROUP BY TO_CHAR(data_comanda,'YYYY')
  HAVING TO_CHAR(data_comanda,'YYYY')=2020
),

```

```

incasari2021 AS (SELECT TO_CHAR(data_comanda,'YYYY'), SUM(cantitate*pret) Suma
  FROM gateste_serveste g JOIN comenzi c ON (g.id_comanda=c.id_comanda)
  JOIN produse p ON (p.id_produs=g.id_produs)
  JOIN angajati a ON (g.id_angajat=a.id_angajat)
  WHERE tip_angajat='ospatar'
  GROUP BY TO_CHAR(data_comanda,'YYYY')

```

```

        HAVING TO_CHAR(data_comanda,'YYYY')=2021
    )
-- s-au calculat incasarile pentru fiecare an cerut, daca se facea o grupare (fara
having - varianta2)
-- exista posibilitatea ca intr-un an cerut sa nu fi existat incasari
SELECT DISTINCT TO_CHAR(data_comanda,'YYYY') An,
DECODE(TO_CHAR(data_comanda,'YYYY'), 2019, (SELECT Suma
                                              FROM incasari2019
                                              ),
2020, (SELECT Suma
                                              FROM incasari2020
                                              ),
2021, (SELECT Suma
                                              FROM incasari2021
                                              ),
0
) Suma_incasari

```

FROM comenzi
ORDER BY An;

--varianta2, fara having

```

SELECT TO_CHAR(data_comanda,'YYYY') An, SUM(cantitate*pret) Suma
FROM gateste_serveste g JOIN comenzi c ON (g.id_comanda=c.id_comanda)
JOIN produse p ON (p.id_produs=g.id_produs)
JOIN angajati a ON (g.id_angajat=a.id_angajat)
WHERE tip_angajat='ospatar'
GROUP BY TO_CHAR(data_comanda,'YYYY')
ORDER BY An;

```

The screenshot shows the Oracle SQL Developer interface. The top part is the 'Query Builder' window with the following code:

```

46 --4. Pentru fiecare an din perioada 2019-2021 sa se afiseze totalul incasarilor restaurantului.
47 --varianta1, folosind having
48
49 WITH incasari2019 AS (SELECT TO_CHAR(data_comanda,'YYYY'), SUM(cantitate*pret) Suma
50   FROM gateste_serveste g JOIN comenzi c ON (g.id_comanda=c.id_comanda)
51           JOIN produse p ON (p.id_produs=g.id_produs)
52           JOIN angajati a ON (g.id_angajat=a.id_angajat)
53 WHERE tip_angajat='ospatar'
54 GROUP BY TO_CHAR(data_comanda,'YYYY')
55 HAVING TO_CHAR(data_comanda,'YYYY')=2019
56 ),
57
58 incasari2020 AS (SELECT TO_CHAR(data_comanda,'YYYY'), SUM(cantitate*pret) Suma
59   FROM gateste_serveste g JOIN comenzi c ON (g.id_comanda=c.id_comanda)
60           JOIN produse p ON (p.id_produs=g.id_produs)
61           JOIN angajati a ON (g.id_angajat=a.id_angajat)
62 WHERE tip_angajat='ospatar'
63 GROUP BY TO_CHAR(data_comanda,'YYYY')
64 HAVING TO_CHAR(data_comanda,'YYYY')=2020
65 ),
66

```

The bottom part is the 'Query Results' window showing the following data:

AN	SUMA_INCASARI
2019	66
2020	373
2021	2522

5. Pentru fiecare comanda se se afiseze daca este o comanda data pentru eveniment, iar in caz contrar sa se afiseze "Fara eveniment", numarul de produse comandate si tipul din care face parte: Comanda mica- pana in 10 produse comandate

- Comanda medie - intre 10 si 30 de produse comandate**
- Comanda mare - peste 30 de produse**

WITH cantitati_comanda AS (SELECT c.id_comanda, SUM(cantitate) Cant --se calculeaza cantitatea pentru fiecare comanda

```
    FROM gateste_serveste g JOIN angajati a ON (g.id_angajat=a.id_angajat)
      JOIN comenzi c ON (c.id_comanda=g.id_comanda)
 WHERE tip_angajat='ospatar'
 GROUP BY c.id_comanda
 )
```

SELECT id_comanda, NVL(TO_CHAR(id_eveniment), 'Fara eveniment') Eveniment, (SELECT
Cant

```
    FROM cantitati_comanda cc
 WHERE cc.id_comanda=c.id_comanda
 ) AS cantitate,
```

CASE --pentru a vedea ce tip de comanda este
(mica/medie/mare)

```
WHEN (SELECT Cant
    FROM cantitati_comanda cc
 WHERE cc.id_comanda=c.id_comanda
 )<10 THEN 'Comanda mica'
WHEN (SELECT Cant
    FROM cantitati_comanda cc
 WHERE cc.id_comanda=c.id_comanda
 )<=30 THEN 'Comanda medie'
ELSE 'Comanda mare'
END AS Tip_Comanda
```

```
FROM comenzi c
ORDER BY id_comanda;
```

```

99
100
101--5. Pentru fiecare comanda se se afiseaza daca este o comanda data pentru eveniment, iar in caz contrar sa se afiseze "Fara eveniment",
102--numarul de produse comandate si tipul din care face parte: Comanda mica- pana in 10 produse comandate
103--Comanda medie - intre 10 si 30 de produse comandate
104--Comanda mare - peste 30 de produse
105
106=WITH cantitati_comanda AS (SELECT c.id_comanda, SUM(cantitate) CANT --se calculeaza cantitatea pentru fiecare comanda
107      FROM gateste_serveste g JOIN angajati a ON (g.id_angajat=a.id_angajat)
108          JOIN comenzi c ON (c.id_comanda=g.id_comanda)
109      WHERE tip_angajat='ospatar'
110      GROUP BY c.id_comanda
111      )
112
113 SELECT id_comanda, NVL(TO_CHAR(id_eveniment),'Fara eveniment') Eveniment, (SELECT CANT
114      FROM cantitati_comanda cc
115      WHERE cc.id_comanda=c.id_comanda
116      ) AS cantitate,
117      CASE --pentru a vedea ce tip de comanda este (mica/medie/mare)
118      WHEN (SELECT CANT
119      FROM cantitati_comanda cc

```

All Rows Fetched: 9 in 0.006 seconds

ID_COMANDA	EVENIMENT	CANTITATE	TIPI_COMANDA
1	10113	58	Comanda mare
2	20 Fara eveniment	2	Comanda mica
3	30 Fara eveniment	8	Comanda mica
4	40110	18	Comanda medie
5	50 Fara eveniment	3	Comanda mica
6	60 Fara eveniment	6	Comanda mica
7	70114	32	Comanda mare
8	80111	44	Comanda mare
9	90112	12	Comanda medie

12. Operații de actualizare și suprimare a datelor

1.Să se mareasca cu 5 lei pretul tuturor produselor din categoria paste.

UPDATE produse

SET pret=pret+5

WHERE id_produs IN (SELECT id_produs

```

        FROM produse p JOIN categorii c ON (p.id_categorie=c.id_categorie)
        WHERE lower(nume_categorie)='paste'
    );

```

SELECT * FROM produse;

ROLLBACK;

The screenshot shows the Oracle SQL Developer interface. The top window is a script editor titled 'ProjectID1.sql' containing SQL code to update products and events. The bottom window is a 'Query Result' viewer showing a table of food items with columns: ID_PRODUS, DENUMIRE, PRET_OBSEVATII, and ID_CATEGORIE.

```

132
133 -- Ex 12: 3 operatii de actualizare sau suprimare a datelor utilizand subcereri
134 -- -1. Sa se mareaasca cu 5 lei pretul tuturor produselor din categoria paste.
135 UPDATE produse
136 SET pret=pret+5
137 WHERE id_produs IN (SELECT id_produs
138   FROM produse p JOIN categorii c ON (p.id_categorie=c.id_categorie)
139   WHERE lower(nume_categorie)='paste'
140 );
141
142 SELECT * FROM produse;
143 ROLLBACK;
144
145 --2. Sa se actualizeze data evenimentului organizat de Popescu Maria din 15-09-2020 in 22-09-2020.
146 UPDATE evenimente
147 SET data_eveniment=TO_DATE('22-09-2020', 'dd-mm-yyyy')
148 WHERE id_eveniment=(SELECT e.id_eveniment
149   FROM evenimente e JOIN comenzi co ON (e.id_eveniment=co.id_eveniment)
150   JOIN clienti c ON (co.id_client=c.id_client)
151   WHERE initcap(nume)='Popescu' AND prenume='Maria' AND
152   data_eveniment=TO_DATE('15-09-2020', 'dd-mm-yyyy')
153 );

```

ID_PRODUS	DENUMIRE	PRET_OBSEVATII	ID_CATEGORIE
1	50paste carbonara	29300q	200
2	51paste bolognese	27 (null)	200
3	52salata cu pui	23200q	210
4	53salata cu boala	200	210
5	54pizza marcherita	17medie	220
6	55pizza capriciosa	19medie	220
7	56tiramisu	15 (null)	230
8	57clatite	14 (null)	230
9	58apa	5500ml	240
10	59suc	7 diverse sortimente	240
11	60vin	10la sticla	240
12	61friptura pui	25230q	250
13	62friptura porc	28200q	250

2. Sa se actualizeze data evenimentului organizat de Popescu Maria din 15-09-2020 in 22-09-2020.

UPDATE evenimente

SET data_eveniment=TO_DATE('22-09-2020', 'dd-mm-yyyy')

WHERE id_eveniment=(SELECT e.id_eveniment

FROM evenimente e JOIN comenzi co ON (e.id_eveniment=co.id_eveniment)
JOIN clienti c ON (co.id_client=c.id_client)

WHERE initcap(nume)='Popescu' AND prenume='Maria' AND
data_eveniment=TO_DATE('15-09-2020', 'dd-mm-yyyy')
);

SELECT * FROM evenimente;

ROLLBACK;

The screenshot shows the Oracle SQL Developer interface. The top window is a 'Query Builder' showing a block of PL/SQL code. The code is as follows:

```

144 --2. Sa se actualizeze data evenimentului organizat de Popescu Maria din 15-09-2020 in 22-09-2020.
145 UPDATE evenimente
146 SET data_eveniment=TO_DATE('22-09-2020', 'dd-mm-yyyy')
147 WHERE id_eveniment=(SELECT e.id_eveniment
148   FROM evenimente e JOIN comenzi co ON (e.id_eveniment=co.id_eveniment)
149     JOIN clienti c ON (co.id_client=c.id_client)
150   WHERE initcap(nume)='Popescu' AND prenume='Maria' AND data_eveniment=TO_DATE('15-09-2020', 'dd-mm-yyyy'))
151 );
152
153
154 SELECT * FROM evenimente;
155 ROLLBACK;
156
157 --3. Sa se stearga comenzile date de clientul cu numele Istrate dupa data de 1 septembrie 2020.
158 DELETE FROM gateste_serveste --intai se sterg liniile din tabelul gateste_serveste deoarece acesta contine fk id_comanda
159 WHERE id_comanda IN (SELECT id_comanda
160   FROM comenzi co JOIN clienti c ON (co.id_client=c.id_client)
161   WHERE initcap(nume)='Istrate' AND data_comanda > TO_DATE('01-09-2020','dd-mm-yyyy'))
162 );

```

The bottom window is a 'Query Result' grid titled 'Script Output' showing the results of the last query:

ID_EVENT	DATA_EVENT	PREFERINTE	ID_TIP_EVENT
1	11013-JUL-21	decor albastru	122
2	11125-JUN-21	(null)	125
3	11222-SEP-20	(null)	124
4	11303-AFR-21	fiori naturale	121
5	11428-FEB-21	decor rosu	123

3. Sa se stearga comenzile date de clientul cu numele Istrate dupa data de 1 septembrie 2020.

DELETE FROM gateste_serveste --intai se sterg liniile din tabelul gateste_serveste deoarece acesta contine fk id_comanda

WHERE id_comanda IN (SELECT id_comanda

```

        FROM comenzi co JOIN clienti c ON (co.id_client=c.id_client)
        WHERE initcap(nume)='Istrate' and data_comanda > TO_DATE('01-09-2020','dd-mm-yyyy')
    );

```

DELETE FROM comenzi --apoi se sterg comenzile ce indeplinesc conditiile mentionate

WHERE id_comanda IN (SELECT id_comanda

```

        FROM comenzi co JOIN clienti c ON (co.id_client=c.id_client)
        WHERE initcap(nume)='Istrate' and data_comanda > TO_DATE('01-09-2020','dd-mm-yyyy')
    );

```

SELECT * FROM gateste_serveste;

SELECT * FROM comenzi;

ROLLBACK;

The screenshot shows the Oracle SQL Developer interface. The top part displays a script in the 'Script' tab:

```

157 ----3. Sa se stearga comenzile date de clientul cu numele Istrate dupa data de 1 septembrie 2020.
158 DELETE FROM gateste_serveste --intai se sterg liniile din tabelul gateste_serveste deoarece acesta contine fk id_comanda
159 WHERE id_comanda IN (SELECT id_comanda
160   FROM comenzi co JOIN clienti c ON (co.id_client=c.id_client)
161   WHERE initcap(nume)='Istrate' and data_comanda > TO_DATE('01-09-2020','dd-mm-yyyy')
162 );
163
164 DELETE FROM comenzi --apoi se sterg comenzile ce indeplinesc conditiile mentionate
165 WHERE id_comanda IN (SELECT id_comanda
166   FROM comenzi co JOIN clienti c ON (co.id_client=c.id_client)
167   WHERE initcap(nume)='Istrate' and data_comanda > TO_DATE('01-09-2020','dd-mm-yyyy')
168 );
169
170 SELECT * FROM gateste_serveste;
171 SELECT * FROM comenzi;
172 ROLLBACK;
    
```

The bottom part shows the 'Query Result' tab with the output of the last two lines of the script:

ID_COMANDA	DATA_COMANDA	ID_MOD_PLATA	ID_CLIENT	ID_EVENTIMENT
1	10 09-MAR-21	600	2	113
2	30 03-MAY-20	620	1	(null)
3	40 05-JUL-21	600	3	110
4	50 08-OCT-19	630	7	(null)
5	60 23-DEC-20	610	4	(null)
6	70 16-FEB-21	620	2	114
7	80 30-APR-21	600	6	111
8	90 02-AUG-20	630	1	112

16. Cerere ce utilizează operația outer-join

Pentru fiecare comanda data dupa data de 1-ianuarie-2020 sa se afiseze numele clientului, data comenzi si tipul de eveniment pentru care a fost data, daca comanda nu este data pentru un eveniment se va afisa "Fara eveniment".

```

SELECT nume, data_comanda, NVL(denumire_tip, 'Fara eveniment') Tip_eveniment
FROM clienti c FULL OUTER JOIN comenzi co ON (c.id_client=co.id_client)
      FULL OUTER JOIN evenimente e ON (co.id_eveniment=e.id_eveniment)
      FULL OUTER JOIN tipuri_evenimente t ON (e.id_tip_eveniment=t.id_tip_eveniment)
WHERE data_comanda>=TO_DATE('01-01-2020', 'dd-mm-yyyy');
    
```

```

File Edit View Navigate Run Source Team Tools Window Help
Connections ProjectDB.sql ProjectDB Log Welcome Page ProjectDB
+ Oracle Connections
  + ID-Serfina
  + ProjectDB
    + Tables (1)
    + Views
    + Indexes
    + Packages
    + Procedures
    + Functions
    + Queues
    + Queues Tab
    + Triggers
    + Types
    + Sequences
    + Materialized
    + Materialized
    + Public Synonyms
    + Database U
    + Public Data
    + Directories
    + Directories
    + Application
    + Java
Reports
  + All Reports
  + Data Dictionary Rep
  + Data Modeler Rep
  + OLAP Reports
  + TimetTen Reports
  + User Defined Reports
  + C:\Users\steff\ProjectDB1.sql
  + Script Output x
  + Query Result x
  + SQL All Rows Fetched: 8 in 0.006 seconds
  1: NAME | DATA_COMANDA | TIP_EVENTIMENT
  2: Radu 09-MAR-21 nuntă
  3: Istrate 15-OCT-20 Fara eveniment
  4: Popescu 03-MAY-20 Fara eveniment
  5: Avram 05-JUL-21 botez
  6: Gavrilă 23-DEC-20 Fara eveniment
  7: Radu 16-FEB-21 aniversare
  8: Comnoiu 30-APR-21 cununie
  9: Popescu 02-AUG-20 revedere

```

Cereri ce utilizeaza operatia division

1. Sa se listeze toti ospatarii care au servit numai produse din categoria bauturi.

WITH id_uri_bauturi AS (SELECT id_produs

```

        FROM produse p JOIN categorii c ON (p.id_categorie=c.id_categorie)
        WHERE lower(nume_categorie)='bauturi'
      )
  
```

SELECT DISTINCT a.id_angajat, nume, prenume --lista cu toti ospatarii care servesc bauturi

FROM angajati a JOIN gateste_serveste g ON (a.id_angajat=g.id_angajat)

WHERE tip_angajat='ospatar' AND id_produs IN (SELECT id_produs

```

          FROM id_uri_bauturi
        )
  
```

MINUS

SELECT DISTINCT a.id_angajat, nume, prenume --lista cu toti ospatarii care servesc alte categorii de produse inafara de bauturi

FROM angajati a JOIN gateste_serveste g ON (a.id_angajat=g.id_angajat)

WHERE tip_angajat='ospatar' AND id_produs NOT IN (SELECT id_produs

```

          FROM id_uri_bauturi
        );
  
```

-- => nu exista ospatari care sa fi servit numai bauturi

```

183 --DIVISION
184 ---1. Sa se listeze toti ospatarii care au servit numai produse din categoria bauturi.
185 WITH id_uri_bauturi AS (SELECT id_produs
186     FROM produse p JOIN categorii c ON (p.id_categorie=c.id_categorie)
187     WHERE lower(nume_categorie)='bauturi'
188 )
189 SELECT DISTINCT a.id_angajat, nume, prenume --lista cu toti ospatarii care servesc bauturi
190 FROM angajati a JOIN gateste_serveste g ON (a.id_angajat=g.id_angajat)
191 WHERE tip_angajat='ospatar' AND id_produs IN (SELECT id_produs
192     FROM id_uri_bauturi
193 )
194 MINUS
195
196 SELECT DISTINCT a.id_angajat, nume, prenume --lista cu toti ospatarii care servesc alte categorii de produse inafara de bauturi
197 FROM angajati a JOIN gateste_serveste g ON (a.id_angajat=g.id_angajat)
198 WHERE tip_angajat='ospatar' AND id_produs NOT IN (SELECT id_produs
199     FROM id_uri_bauturi
200 );
201
202 --- => nu exista ospatari care sa fi servit numai bauturi
203
204

```

2. Sa se obtina toti bucatarii care au gatit pentru cel putin doua evenimente.

```

SELECT a.id_angajat, nume
FROM angajati a JOIN gateste_serveste g ON (a.id_angajat=g.id_angajat)
WHERE tip_angajat='bucatar' AND g.id_comanda IN
    (SELECT id_comanda
     FROM comenzi
     WHERE id_eveniment IS NOT NULL
    ) --sunt selectati toti bucatarii care au gatit pentru evenimente
GROUP BY a.id_angajat, nume
HAVING 2<= (SELECT COUNT(*) --este pusa conditia sa fi gatit la cel putin doua evenimente
            FROM gateste_serveste gs
            WHERE gs.id_angajat=a.id_angajat AND gs.id_comanda IN
                (SELECT id_comanda
                 FROM comenzi
                 WHERE id_eveniment IS NOT NULL
                )
           GROUP BY gs.id_angajat
          );

```

The screenshot shows the Oracle SQL Developer interface. The top window displays a complex SQL query (lines 204 to 233) designed to find employees who have worked on at least two different events. The bottom window shows the execution results, which are two rows of data:

ID_ANGAJAT	NUME
206	Brezan
204	Dinu

17. Optimizarea unei cereri

Sa se listeze toti furnizorii (id-ul si denumirea) care au furnizat ingrediente ce au fost continute de produsele din comanda clientilor cu numele "Popescu" si sunt comenzi date pentru evenimente.

Cererea SQL:

--var1 (6 join-uri)

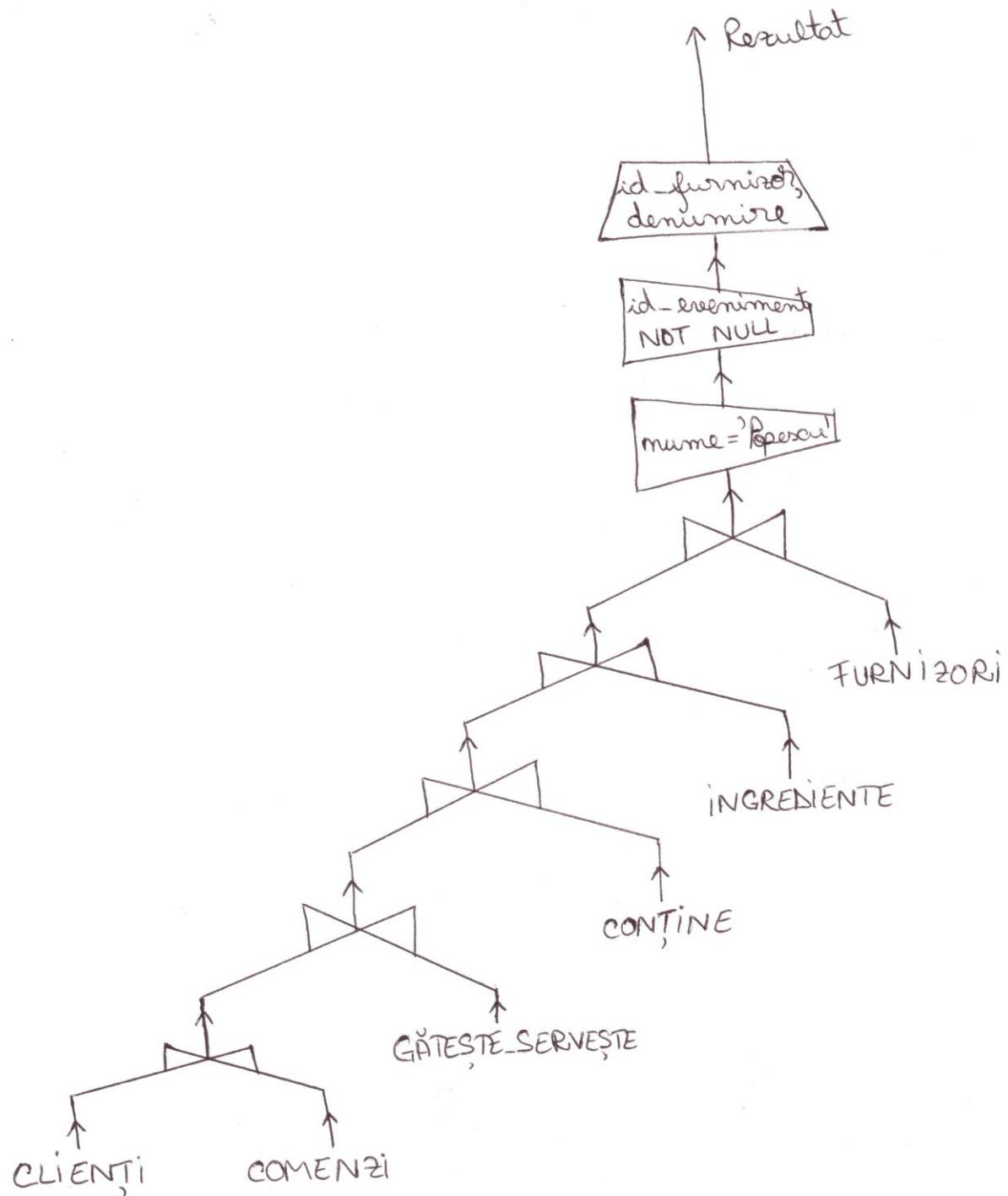
```
SELECT DISTINCT f.id_furnizor, f.denumire
FROM furnizori f JOIN ingrediente i ON (f.id_furnizor=i.id_furnizor)
                  JOIN contine c ON (i.id_ingredient=c.id_ingredient)
                  JOIN produse p ON (p.id_produs=c.id_produs)
                  JOIN gateste_serveste g ON (p.id_produs=g.id_produs)
                  JOIN comenzi co ON (g.id_comanda=co.id_comanda)
                  JOIN clienti cli ON (cli.id_client=co.id_client)
WHERE UPPER(cli.nume)='POPESCU' AND co.id_eveniment IS NOT NULL;
```

--var2

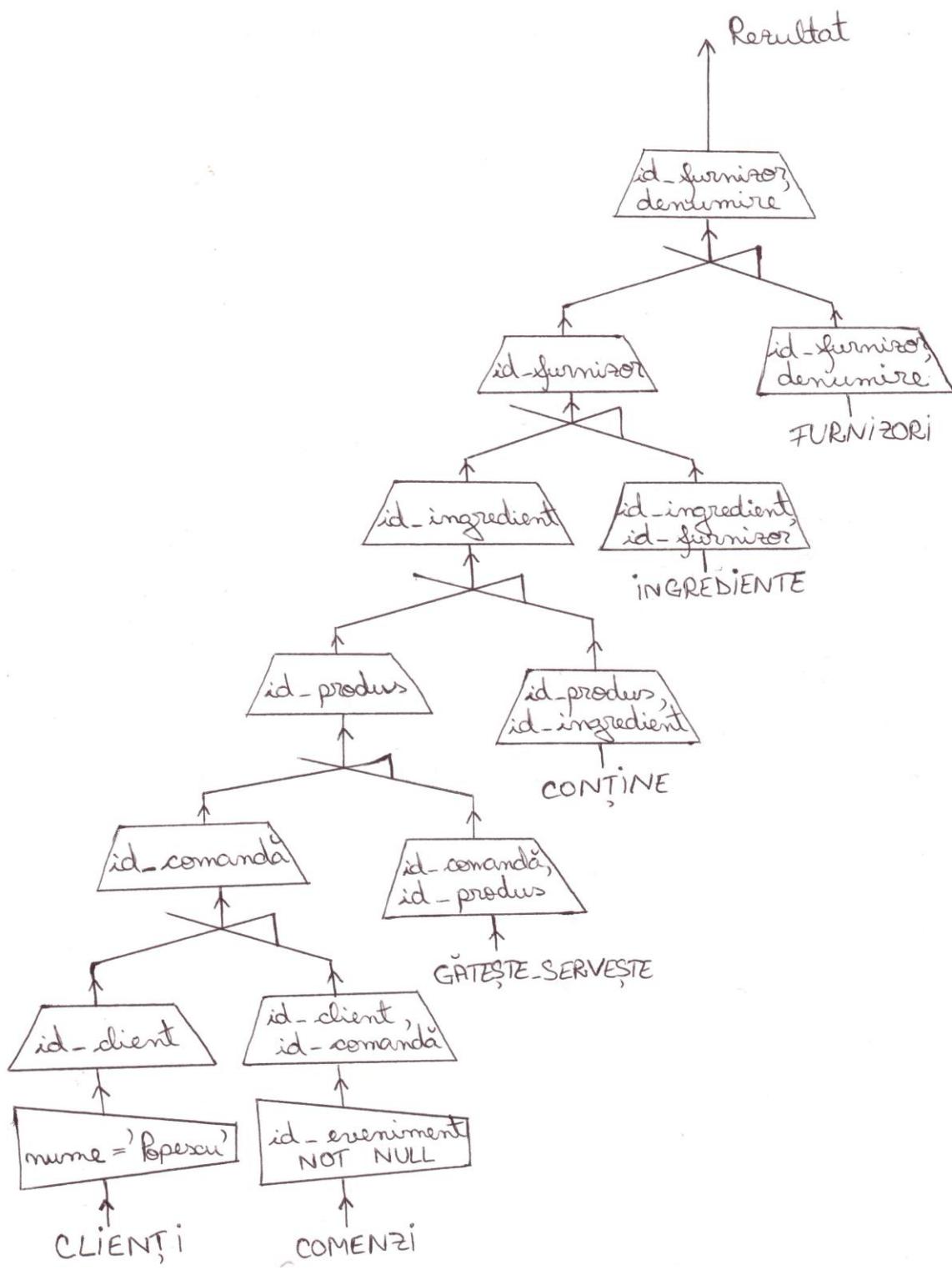
```
SELECT DISTINCT f.id_furnizor, f.denumire
FROM clienti cli JOIN comenzi co ON (cli.id_client=co.id_client)
                  JOIN gateste_serveste g ON (g.id_comanda=co.id_comanda)
                  JOIN contine c ON (c.id_produs=g.id_produs)
                  JOIN ingrediente i ON (i.id_ingredient=c.id_ingredient)
                  JOIN furnizori f ON (i.id_furnizor=f.id_furnizor)
WHERE UPPER(cli.nume)='POPESCU' AND co.id_eveniment IS NOT NULL;
```

În urma proiectării arborelui s-a observat ca JOIN-ul cu tabelul PRODUSE nu schimbă rezultatul, deoarece id_produs care se găsește în tabelele GĂTESTE_SERVESTE și CONTINE referă cheia primara id_produs din tabelul PRODUSE. De aceea, acest JOIN a fost eliminat pentru a obține un arbore mai optim.

Arborele algebric anterior optimizării:



Arborele algebric ulterior optimizării:



Selectiile au fost efectuate cât mai devreme posibil pentru a reduce dimensiunea relațiilor. Proiecțiile au fost executate la început și între fiecare join pentru a elimina attributele care nu sunt folosite în operațiile următoare. Au fost folosite proprietățile de comutare a selecției cu join-ul și de comutare a proiecției cu join-ul.

Expresia algebraica:

```
R1 = SELECT ( CLIENTI, nume='Popescu' )
R2 = PROJECT ( R1, id_client )
R3 = SELECT ( COMENZI, id_eveniment NOT NULL )
R4 = PROJECT ( R2, id_client, id_comanda )
R5 = SEMIJOIN( R2, R4, id_client )
R6 = PROJECT ( R5, id_comanda )
R7 = PROJECT ( GATESTE_SERVESTE, id_comanda, id_produs )
R8 = SEMIJOIN ( R6, R7 )
R9 = PROJECT ( R8, id_produs )
R10 = PROJECT ( CONTINE, id_produs, id_ingredient )
R11 = SEMIJOIN ( R9, R10 )
R12 = PROJECT ( R11, id_ingredient )
R13 = PROJECT ( INGREDIENTE, id_ingredient, id_furnizor )
R14 = SEMIJOIN ( R12, R13 )
R15 = PROJECT ( R14, id_furnizor )
R16 = PROJECT ( FURNIZORI, id_furnizor, denumire )
R17 = SEMIJOIN ( R15, R16 )
Rezultat = PROJECT ( R17, id_furnizor, denumire )
```

18. BCNF, FN4, FN5 și denormalizare

BCNF

Ca o relație să fie în BCNF trebuie ca fiecare determinant să fie o cheie candidat.

Presupunem că un anumit tip de produs poate fi gătit de un singur bucătar.

Înțial tabelul GATESTE_SERVESTE arată în felul următor:

GATESTE_SERVESTE			
id_comanda#	id_angajat#	id_produs	cantitate
10	200	51	2
20	202	62	5
30	200	51	4
40	204	57	3
20	204	57	2

{id_comanda#, id_angajat#} -> {id_produs}

{id_produs} -> {id_angajat} (atributul id_produs este o cheie candidat deoarece un produs poate fi gătit de un singur bucătar, deci apare o singură dată alături de bucătarul corespunzător)

Aplicăm regula Casey-Delobel:

R1 (id_produs#, id_angajat) – cheia candidat a devenit cheie primară

R2 (id_comandă#, id_produs#, cantitate)

După realizarea normalizării în BCNF, tabelele arată în felul următor:

id_produs#	id_angajat
51	200
62	202
51	200
57	204
57	204

id_comanda#	id_produs#	cantitate
10	51	2
20	62	5
30	51	4
40	57	3
20	57	2

Astfel, fiecare determinant este o cheie candidat, iar, acum, relațiile rezultate se află în BCNF.

FN4

Ca o relație să fie în FN4 trebuie ca aceasta să fie în BCNF și să nu conțină relații m:n independente.

Presupunem că avem un lanț de restaurante, fiecare punând la dispoziție clienților produse din anumite categorii (acestea pot coincide, dar pot fi și diferite), iar acestea pot fi servite fiecare de ospătarii angajați în restaurantul respectiv.

Exemplu:

Restaurantul R1 oferă pizza și paste care pot fi servite de ospătarii cu id-urile 201 și 203.

Restaurantul R2 oferă pizza și băuturi care pot fi servite de ospătarul cu id-ul 205.

restaurant#	categorie#	id_ospatar#
R1	pizza	201
R1	pizza	203
R1	paste	201
R1	paste	203
R2	pizza	205
R2	băuturi	205

Restaurant → → categorie

Restaurant → → ospatar

Relația de mai sus se va descompune în felul următor:

R1 (restaurant#, categorie#)

R2 (restaurant#, ospatar#)

restaurant#	categorie#
R1	pizza
R1	paste
R2	pizza
R2	bauturi

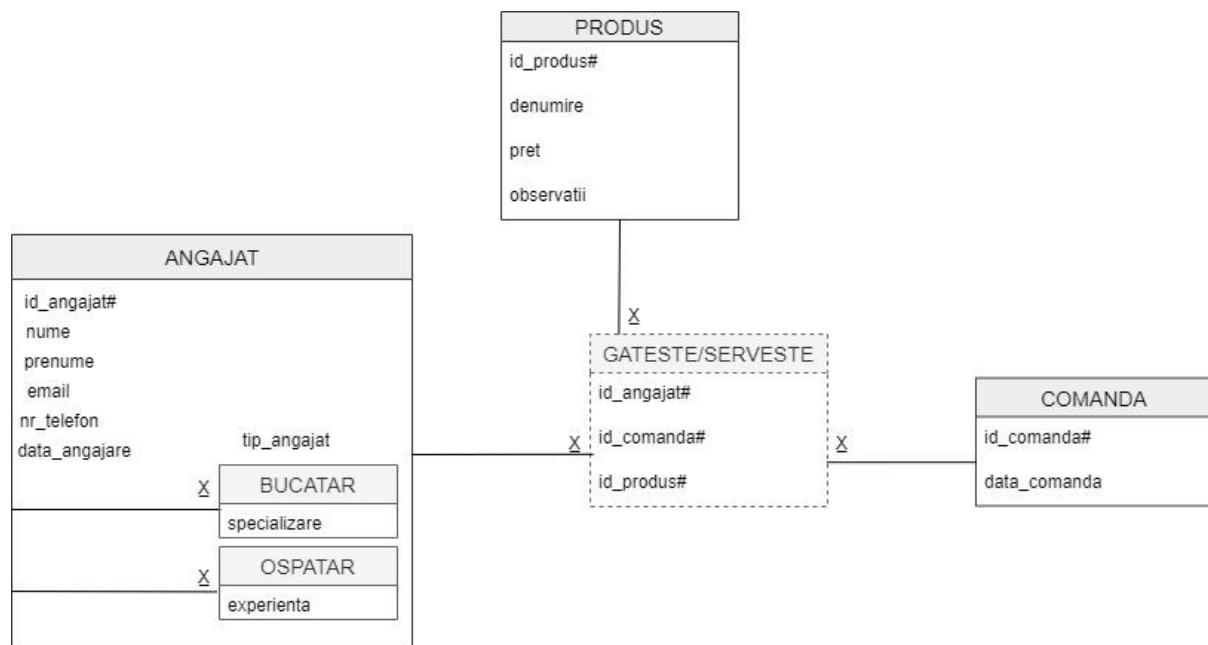
restaurant#	id_ospatar#
R1	201
R1	203
R2	205

Astfel, au fost eliminate dependențele multiple, iar, acum, relațiile rezultate se află în FN4.

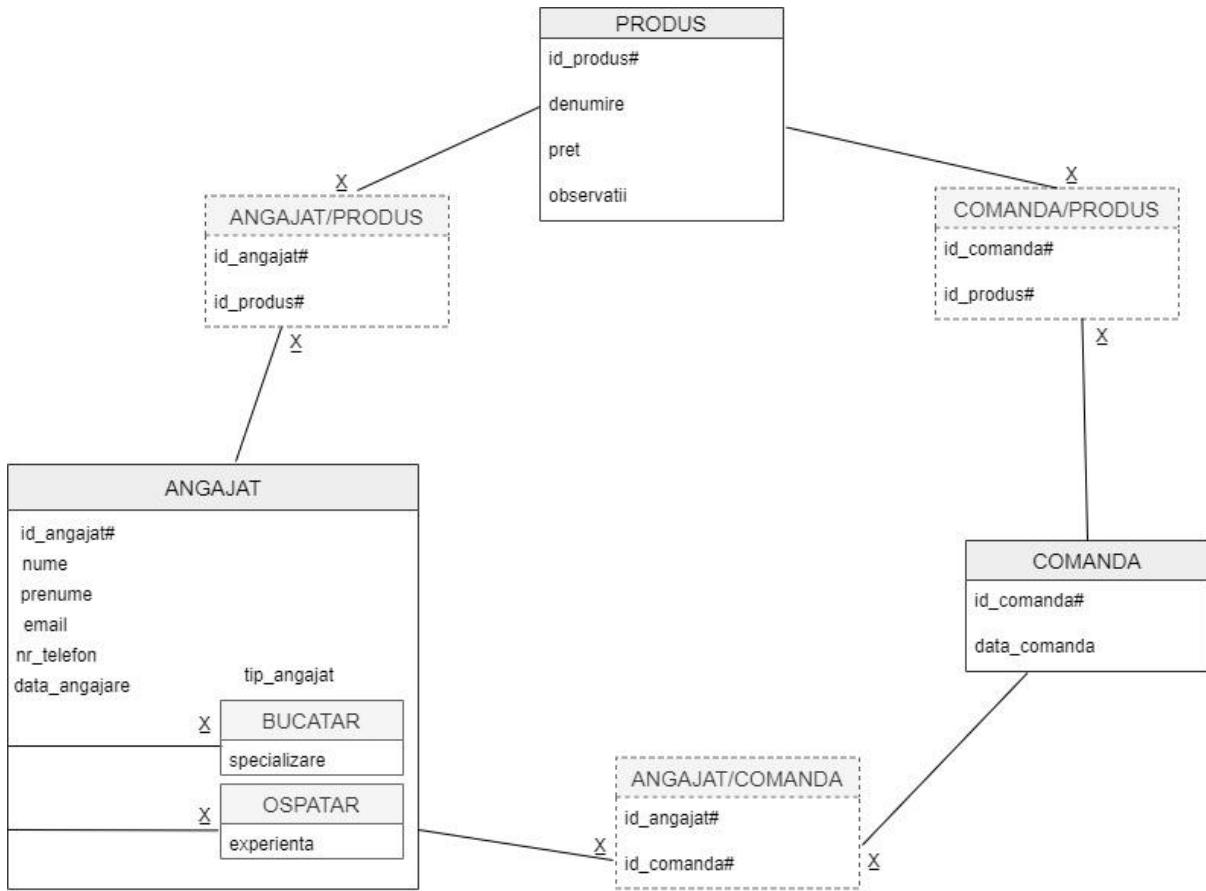
FN5

Ca o relație să fie în FN5 trebuie să fie eliminate redundanțele care apar în relații m:n dependente (+ sa fie în FN4).

O comandă poate contine mai multe produse care pot fi găsite/servite de mai mulți angajați.



Această relație de tip 3 poate fi echivalentă cu 3 relații de tip 2 doar dacă aceste relații de tip 2 sunt ciclice.



Relația este ciclică, iar atunci când facem toate join-urile vom obține un rezultat echivalent cu cel obținut din relația de tip 3. O relație pentru a fi în FN5 trebuie să nu conțină dependențe ciclice. Se observă că cele 3 relații de tip 2 compun o diagramă care conține dependențe ciclice, deci relația de mai sus nu se află în FN5.

=>În schimb, relația de tip 3 se află în FN5.

Denormalizare

Denormalizarea este necesară, deoarece constă în reducerea numărului de join-uri care trebuie efectuate pentru rezolvarea unei interogări, prin realizarea unora dintre acestea în avans, ca făcând parte din proiectarea bazei de date. Astfel este mărită redundanța cu scopul de a micșora timpul de execuție.

Să presupunem că în tabelul ANGAJATI se află numai ospătari și există numai atrbutele id_angajat, nume, prenume și specializare. De exemplu:

id_angajat	nume	prenume	specializare
201	Dumitrache	Bianca	Franța
203	Nistor	Izabela	Spania
205	Ionescu	Ion	Franța

Se observă că pe coloana specializare pot fi valori repetitive, mai mulți angajați pot face specializarea în aceeași țară. Presupunem că în baza de date ar exista un tabel separat în care s-ar afla specializarea și id-ul angajatului corespunzător. În acest caz, este necesar procesul de denormalizare în urma căruia atributul greutate se va plasa în tabelul ANGAJATI, deoarece nu este eficient ca acest atribut să se afle într-un tabel separat.