**Gestionarea unei pizzerii**

**BAZE DE DATE**

**Radulescu Alexia-Bianca**

**Grupa 133**

Ex1. Descrierea modelului real, a utilității acestuia și a regulilor de funcționare.

Baza de date contine informatii atat despre angajatii pizzeriei cat si despre clientii sai. Clientii au optiunea de a comanda online sau direct in restaurant unde pot sa ocupe o masa, pentru a se bucura de mancarea comandata, sau pot lua mancarea la pachet. Clientii pot alege sa comande pizza si/sau bauturi si/sau desert.

La orice pizza comandata clientul are de ales dintre mai multe dimensiuni, tipuri de blat si sosuri si are optiunea de a adauga topinguri. Pentru fiecare comanda este emisa o nota de plata care poate fi platita cu cash sau cu card. Comenzile comandate online sunt livrate de un angajat al pizzeriei care are jobul de curier.

La fiecare comanda exista un angajat care are rolul de a prelua cererile clientilor si a le transmite bucatariei. Iar fiecare pizza preparata are asociat un bucatar.

Aceasta baza de date are rolul de a tine evidenta clientilor, comenzilor si banilor incasati prin fiecare comanda, dar si angajatilor si participarii lor in fiecare comanda.

Ex2. Prezentarea constrângerilor (restricții, reguli) impuse asupra modelului.

Modelul are urmatoarele constrangeri:

Un angajat poate avea un singur job.

O comanda poate fi primita de un singur angajat, dar un angajat poate primii mai multe comenzi.

O pizza poate fi pregatita de un singur angajat, iar un angajat poate pregati cel mult o pizza

De asemenea, o comanda poate fi livrata de un singur angajat, dar un angajat poate livra mai multe comenzi.

Un client poate face mai multe comenzi, dar o comanda poate fi facuta de cel mult un client.

Un client care comanda online poate primii mancarea intr-o singura locatie.

Un client care doreste sa serveasca mancarea in restaurant se poate aseza la o singura masa, dar mai multi clienti se pot aseza la o masa.

O comanda poate contine mai multe pizze si mai multe bauturi si/sau dulciuri.

Aceeasi pizza poate face parte din mai multe comenzi.

Mai multe pizze pot avea acelasi tip blat, dar o pizza poate avea un singur tip de blat.

O pizza poate fi servita cu mai multe sosuri si un sos poate fi servit cu mai multe pizze.

La o pizza pot fi adaugate mai multe topinguri, si un toping poate fi adaugat la mai multe pizze.

O pizza poate avea o singura dimensiune, dar mai multe pizze pot avea aceeasi dimensiune.

Aceeasi bautura si desert poate face parte din mai multe comenzi.

O comanda are o singura nota de plata care poate avea o singura metoda de plata.

Ex3. Descrierea entităților, incluzând precizarea cheii primare.

Baza de date are urmatoarele entitati: CUSTOMERS, ONLINE\_CUSTOMERS, IN\_CUSTOMERS, TO\_GO\_CUSTOMERS, LOCATIONS, EMPLOYEES, JOBS, DELIVERIES, TABLES, ORDERS, RECEIPTS, PAYMENT\_METHODS, PIZZA, SIDES, DRINKS, SWEETS, CRUSTS, SOUCES, EXTRA\_TOPPINGS, SIZES, CONTINE, SE\_SERVESTE\_CU, ARE\_IN\_PLUS.

Voi incepe prin prezentarea entitatilor independente.

CUSTOMERS= persoana fizica, client al pizzeriei, care poate face comenzi de pizza cu sau fara bauturi si desert, care poate plasa comanda fizic pentru a manca la restaurant sau pentru a lua mancarea la pachet, dar si online pentru a primii mancarea prin livrare intr-o locatie aleasa de el. CUSTOMERS este superentitate pentru ONLINE\_CUSTOMERS, IN\_CUSTOMERS si TO\_GO\_CUSTOMERS care reprezinta subentitati. Cheia primara este customer\_id.

ONLINE\_CUSTOMERS= persoana fizica, client al pizzeriei, care plaseaza comenzi online. Acesta trebuie sa ofere date precum numele sau, numarul de telefon, mailul si adresa, pentru a informa curuierul unde doreste sa fie livrata mancarea si pentru a putea fi contactat de curier in momentul livrarii. Cheia primara este customer\_id.

IN\_CUSTOMERS= persoana fizica, client al pizzeriei, care plaseaza comenzi fizic si doreste sa serveasca mancarea direct in restaurant. Clientul se aseaza la o masa libera. Cheia primara este customer\_id.

TO\_GO\_CUSTOMERS= persoana fizica, client al pizzeriei, care plaseaza comanda fizic, dar doreste sa primeasca mancarea la pachet in loc sa manance in restaurant. Cheia primara este customer\_id.

LOCATIONS= entitatea utilizata pentru a accesa detalii despre locatiile unde locuiesc angajatii si locatiile in care doresc clientii care comanda online sa le fie livrata mancarea. Cheia primara este location\_id.

EMPLOYEES= persoana fizica angajat al pizzeriei, care poate avea un singur job in cadrul pizzeriei. In cazul in care angajatul are jobul de chelner atunci el poate preluamai multe comenzi in acelasi timp. Daca are jobul de curier e poate livra mai multe comenzi in acelasi timp. Iar daca are jobul de bucatar, poate pregati mai multe pizze in acelasi timp. Cheia primara este employee\_id.

JOBS= entitate utilizata pentru a tine evidenta tuturor joburilor existente in cadrl pizzeriei. Prezinta si informatii despre salariul minim si maxim pe care il poate avea un angajat. Cheia primara este job \_id.

DELIVERIES= entitate utilizata pentru a tine evidenta tuturor livrarilor pizzeriei, angajatilor care au efectuat livrarile si cat timp a durat fiecare livrare. Cheia primara este delivery\_id.

TABLES= etitate utilizata pentru a tine evidenta tuturor meselor si numarul de locuri. Cheia primara este table\_id.

ORDERS= etitate utilizata pentru a tine evidenta tuturor comenzilor effectuate, cu detalii despre cee ace a fost comandat, detalii despre clientul care a plasat comanda si despre angajatul care a preluat-o. Cheia primara este order\_id.

RECEIPTS= etitate utilizata pentru a tine evidenta notelor de plata effectuate pentru fiecare comanda. Cheia primara este receipt\_id.

PAYMENT\_METHODS= etitate utilizata pentru a tine evidenta platilor si a tipului de plata ales de client, cash sau card. Cheia primara este payment \_id.

PIZZA= contine informatii despre fiecare pizza pregatita si pretul ei. O pizza poate avea un singur tip de blat, o singura dimensiune, mai multe topinguri si poate fi servita cu mai multe sosuri. Cheia primara este pizza\_id.

SIDES= contine informatii despre celelalte alimente pe care clientii doresc sa le comande pe langa pizza, cum ar fi bauturi si desert, numele acestor alimente si pretul lor. SIDES este superentitate pentru SWEETS si DRINKS care reprezinta subentitati. Cheia primara este side\_id.

SWEETS= contine informatii despre deserturile comandate de clienti. Cheia primara este side\_id.

DRINKS = contine informatii despre bauturile comandate de client. Cheia primara este side\_id.

CRUSTS= contine informatii despre tipurile de blaturi de pizza oferite de pizzerie si pretul fiecaruia. Cheia primara este crust \_id.

SOUCES= contine informatii despre sosurile pentru pizza oferite de pizzeria si pretul fiecaruia. Cheia primara este souce\_id.

EXTRA\_TOPPINGS= contine informatii despre topingurile oferite de pizzerie care pot fi adaugate la o pizza si pretul fiecaruia. Cheia primara este topping\_id.

SIZES= contine informatii despre dimensiunile de pizza oferite de pizzeria si pretul fiecaruia. Cheia primara este size\_id.

Voi prezenta si entitatile dependente.

CONTINE= cuprinde informatii despre pizzele pregatite, alimente vandute si comenzile din care acestea fac parte. Cheia primara este compusa din pizza\_id, order\_id si side\_id.

SE\_SERVESTE\_CU= cuprinde informatii despre pizzele pregatite, si sosurile comandate impreuna cu acestea. Cheia primara este compusa din pizza\_id si souce\_id.

ARE\_IN\_PUS= cuprinde informatii despre pizzele pregatite, si topingurile adaugate de client. Cheia primara este compusa din pizza\_id si topping\_id.

Ex4. Descrierea relațiilor, incluzând precizarea cardinalității acestora.

Voi prezenta reatiile din baza de date folosind urmatoarea denumire: ENTITATE\_relatie\_ENTITATE.

CUSTOMERS\_comanda\_ORDERS= relatie carea leaga clientii de comenzile lor. Cardinalitatea minima 0:1 si cardinalitatea maxima este 1:M.

ONLINE\_CUSTOMERS\_primeste\_DELIVERIES= relatie care leaga clientii care au plasat o comanda online de livrarea comenzilor. Cardinalitatea minima 1:1 si cardinalitatea maxima este 1:1.

ONLINE\_CUSTOMERS\_cere mancarea in locatia\_ LOCATIONS= relatie care leaga clientii care au plasat o comanda online de locatia in care doresc sa le fie livrata mancarea. Cardinalitatea minima 0:1 si cardinalitatea maxima este M:1.

IN\_CUSTOMERS\_se aseaza\_TABLES= relatie care leaga clientii care decid sa comande fizic si sa manance in restaurant de masa la care se aseaza. Cardinalitatea minima 0:1 si cardinalitatea maxima este 1:1.

EMPLOYEES\_livreaza\_DELIVERIES= relatie care leaga angajatii de livrarile pe care le efectueaza. Cardinalitatea minima 1:0 si cardinalitatea maxima este 1:M.

EMPLOYEES\_efectueaza\_ORDERS= relatie care leaga angajatii de comenzile pe care le-au preluat de la client. Cardinalitatea minima 1:0 si cardinalitatea maxima este 1:M.

EMPLOYEES\_locuieste\_LOCATIONS= relatie care leaga angajatii de locul in care locuiesc, informatie utilizata in descrierea angajatului. Cardinalitatea este 1:1.

EMPLOYEES\_are\_JOBS= relatie care leaga angajatul de jobul pe care il are in cadrul pizzeriei. Cardinalitatea este 1:1.

EMPLOYEES pregateste PIZZA= relatie care leaga pizza de bucatarul sau asistentul de bucatar care o pregateste. Cardinalitatea este 1:M(0).

ORDERS\_costa\_RECEIPTS= relatie care leaga comanda cu nota sa de plata. Cardinalitatea minima 1:1 si cardinalitatea maxima este M:1.

RECEIPTS\_plateste\_PAYMENT\_METHODS= relatie care leaga nota de plata de modalitatea in care aceasta a fost platita, adica cu cash sau cu card. Cardinalitatea este 1:1.

ORDERS\_contine\_PIZZA contine SIDES= relatie ternara care leaga pizzele si alimentele precum bauturi si deserturi de comenzile din care fac parte. Cardinalitatea minima 0:0 si cardinalitatea maxima este M:M.

PIZZA\_are\_CRUSTS= relatie care leaga pizza de tipul de blat ales de client. Cardinalitatea minima 0:1 si cardinalitatea maxima este M:1.

PIZZA\_se serveste cu\_SOUCES= relatie care leaga pizza de sosul cu care clientul alege sa o serveasca. Cardinalitatea minima 0:0 si cardinalitatea maxima este M:M.

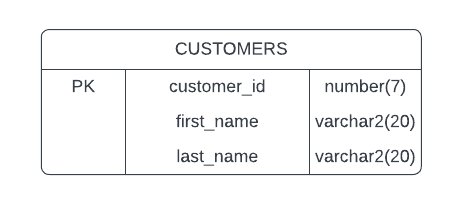
PIZZA\_are in plus\_EXTRA\_TOPPINGS= relatie care leaga pizza de topingurile pe care clientul le adauga. Cardinalitatea minima 0:0 si cardinalitatea maxima este M:M.

PIZZA\_are\_SIZES= relatie care leaga pizza de dimensiunea pe care o doreste clientul. Cardinalitatea minima 0:1 si cardinalitatea maxima este M:1.

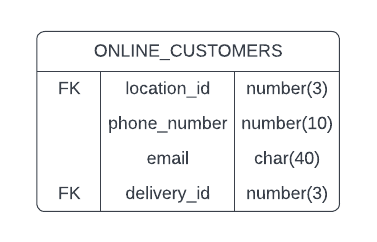
EX5. Descrierea atributelor, incluzând tipul de date și eventualele constrângeri, valori implicite, valori posibile ale atributelor.

Superentitatea CUSTOMERS are urmatoarele atribute:

* customer\_id= variabila de tip intreg, de lungime 7, care reprezinta codul clientului si este cheie primara;
* first\_name= variabila de tip caracter, de lungime maxima 20, care reprezinta prenumele clientului, NOT NULL;
* last\_name= variabila de tip caracter, de lungime maxima 20, care reprezinta numele de familie al clientului, NOT NULL;

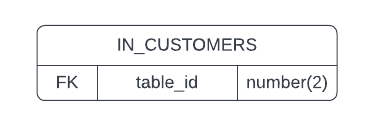


Subentitatea ONLINE\_CUSTOMERS are urmatoarele atribute:

* customer\_id= variabila de tip intreg, de lungime 7, care reprezinta codul clientului, este cheie primara;
* first\_name= variabila de tip caracter, de lungime maxima 20, care reprezinta prenumele clientului, NOT NULL;
* last\_name= variabila de tip caracter, de lungime maxima 20, care reprezinta numele de familie al clientului, NOT NULL;
* location\_id= variabila de tip intreg, de lungime 3, care reprezinta codul locatiei in care clientul vrea sa fie livrata comanda, este cheie secundara;
* phone\_number= variabila de tip intreg, de lungime 10, care reprezinta numarul de telefon prin care poate fi contactat clientul in momentul livrarii mancarii, UNIQUE;
* email= variabila de tip caracter, de lungima maxima 40, care reprezinta emailul prin care poate fi contactat clientul pentru a primii notificari si oferte ale pizzeriei, UNIQUE;
* delivery\_id= variabila de tip intreg, de lungime 3, care reprezinta codul livrarii prin care este trimisa comanda catre client, este cheie secundara; 

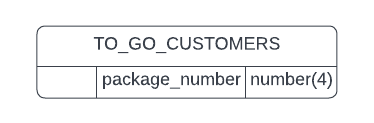
Subentitatea IN\_CUSTOMERS are urmatoarele atribute:

* customer\_id= variabila de tip intreg, de lungime 7, care reprezinta codul clientului si este cheie primara;
* first\_name= variabila de tip caracter, de lungime maxima 20, care reprezinta prenumele clientului, NOT NULL;
* last\_name= variabila de tip caracter, de lungime maxima 20, care reprezinta numele de familie al clientului, NOT NULL;
* table\_id= variabila de tip intreg, de lungime 2, care reprezinta codul mesei la care se aseaza clientul, NOT NULL;



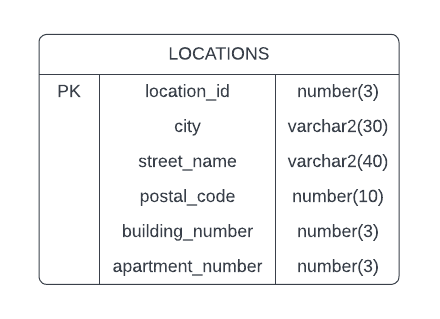
Subentitatea TO\_GO\_CUSTOMERS are urmatoarele atribute:

* customer\_id= variabila de tip intreg, de lungime 3, care reprezinta codul clientului si este cheie primara;
* first\_name= variabila de tip caracter, de lungime maxima 20, care reprezinta prenumele clientului, NOT NULL;
* last\_name= variabila de tip caracter, de lungime maxima 20, care reprezinta numele de familie al clientului, NOT NULL;
* package\_number= variabila de tip intreg, de lungime maxima 4, care reprezinta numarul pachetului comandat de client, NOT NULL;



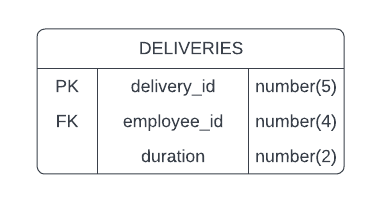
Entitatea LOCATIONS are urmatoarele atribute:

* location\_id= variabila de tip intreg, de lungime 3, care reprezinta codul locatiei, este cheie primara;
* city = variabila de tip caracter, de lungime maxima 30, care reprezinta orasul, NOT NULL;
* street\_name= variabila de tip caracter, de lungime maxima 40, care reprezinta numele strazii, NOT NULL;
* postal\_code= variabila de tip caracter, de lungime maxima 10, care reprezinta codul postal, NOT NULL;
* building\_number= variabila de tip caracter, de lungime maxima 3, care reprezintanumarul cadirii;
* apartment\_number= variabila de tip caracter, de lungime maxima 3, care reprezinta numarul apartamentului;



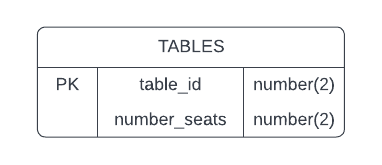
Entitatea DELIVERIES are urmatoarele atribute:

* delivery\_id= variabila de tip intreg, de lungime 5, care reprezinta codul livrarii comenzii, este cheie primara;
* employee\_id= variabila de tip intreg, de lungime 4, care reprezinta codul angajatului care face livrarea, este cheie externa;
* duration= variabila de tip intreg, de lungime 2, care reprezinta cat timp i\_a luat curierului sa faca livrarea, NOT NULL;

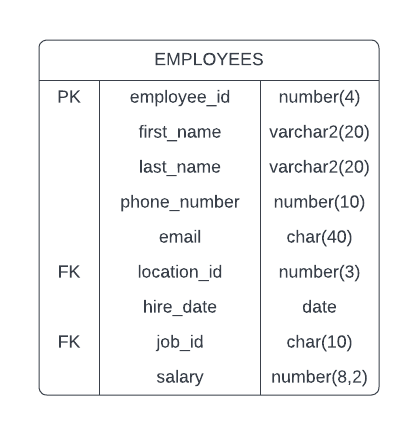


Entitatea TABLES are urmatoarele atribute:

* table\_id= variabila de tip intreg, de lungime 2, care reprezinta numarul mesei, este cheie primara;
* number\_seats= variabila de tip intreg, de lungime 2, care reprezinta numarul de locuri pe care le are masa, NOT NULL;

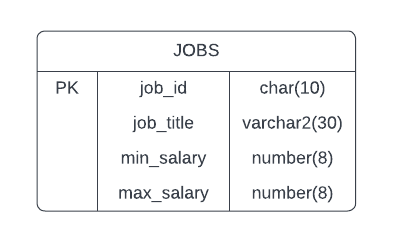


Entitatea EMPLOYEES are urmatoarele atribute:

* employee\_id= variabila de tip intreg, de lungime 4, care reprezinta codul angajatului, este cheie primara;
* first\_name= variabila de tip caracter, de lungime maxima 20, care reprezinta prenumele angajatului, NOT NULL;
* last\_name= variabila de tip caracter, de lungime maxima 20, care reprezinta numele de familie angajatului, NOT NULL;
* phone\_number= variabila de tip intreg, de lungime 10, care reprezinta numarul de telefon al angajatului, UNIQUE;
* email= variabila de tip caracter, de lungime maxima 40, care reprezinta emailul angajatului, UNIQUE;
* location\_id= variabila de tip intreg, de lungime 3, care reprezinta adresa angajatului, cheie secundara;
* hire\_date= variabila de tip data, care reprezinta data in care a fost angajat salariatul, NOT NULL;
* job\_id= variabila de tip caracter, de lungime maxima 10, care reprezinta codul jobului pe care il are angajatul, cheie secundara;
* salary= variabila de tip intreg, de lungime maxima 8 cu maxim 2 zecimale, NOT NU

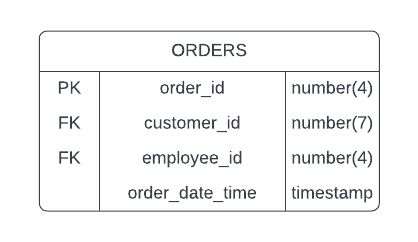
Entitatea JOBS are urmatoarele atribute:

* job\_id= variabila de tip caracter, de lungime maxima 10, care reprezinta codul jobului, este cheie principala;
* job\_title= variabila de tip caracter, de lungime maxima 30, care reprezinta denumirea jobului, NOT NULL;
* min\_salary= variabila de tip intreg, de lungime maxima 8, care reprezinta salariul minim pe care il poate avea jobul, NOT NULL;
* max\_salary= variabila de tip intreg, de lungime maxima 8, care reprezinta salariul maxim pe care il poate avea jobul, NOT NULL;



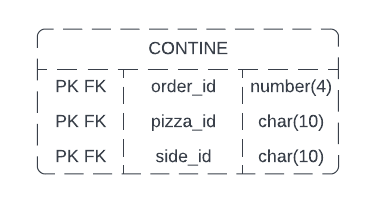
Entitatea ORDERS are urmatoarele atribute:

* order\_id= variabila de tip intreg, de lungime 4, care reprezinta codul comenzii, este cheie primara;
* customer\_id= variabila de tip intreg, de lungime 7, care reprezinta codul clientului care a plasat comanda, este cheie secundara;
* employee\_id= variabila de tip intreg, de lungime 4, care reprezinta codul angajatului care a preluat comanda, este cheie secundara;
* order\_date\_time= variabila de tip timestamp, care reprezinta data si timpul la care clientul a plasat comanda, NOT NULL;



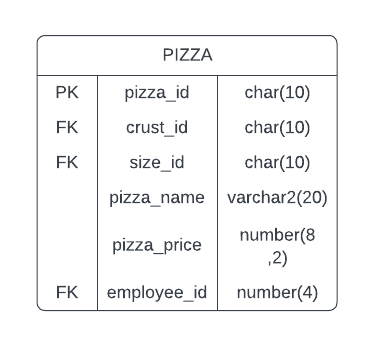
Relatia ORDERS\_contine\_PIZZA\_contine\_SIDE=> entitatea CONTINE are urmatoarele atribute:

* order\_id= variabila de tip intreg, de lungime 4, care reprezinta codul comenzii, atribut care corespunde valorii unei chei primare din ORDERS;
* pizza\_id= variabila de tipcaracter, de lungime maxima 10, care reprezinta codul pizzei, atribut care corespunde valorii unei chei primare din PIZZA;
* side\_id= variabila de tip caracter, de lungime maxima 10, care reprezinta codul alimentului diferit de pizza, atribut care corespunde valorii unei chei primare din SIDE;



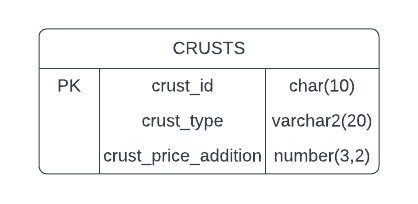
Entitatea PIZZA are urmatoarele atribute:

* pizza\_id= variabila de tip caracter, de lungime maxma 10, care reprezinta codul pizzei, este cheie primara;
* crust\_id= variabila de tip caracter, de lungime maxima 10, care reprezinta codul tipului de blat al pizzei, este cheie secundara;
* size\_id= variabila de tip caracter, de lungime maxima 10, care reprezinta codul dimensiunii pizzei, este cheie secundara;
* pizza\_name= vaiabila de tip caracter, de lugime maxima 20, care reprezinta denumirea pizzeria, NOT NULL;
* pizza\_price= variabila de tip intreg, de lungime maxima 8 cu maxim 2 zecimale, NOT NULL;
* employee\_id= variabila de tip intreg, de lungime 4, care reprezinta codul angajatului care prepara pizza, este cheie secundara;



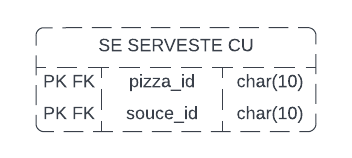
Entitatea CRUSTS are urmatoarele atribute:

* curst\_id= variabila de tip caracter, de lungime maxima 10, care reprezinta codul tipului de blat, este cheie primara;
* crust\_type= variabila de tip caracter, de lungime maxima 20, care reprezinta denumirea tipului de blat, NOT NULL;
* crust\_price\_addition= variabila de tip intreg, de lungime maxima 3 cu maxim 2 zecimale, care reprezinta pretul tipului de blat;



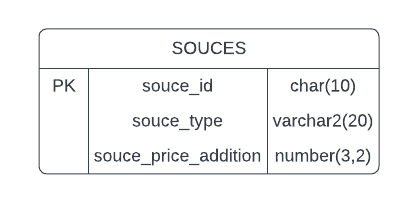
Relatia PIZZA\_se serveste cu\_SOUCES=> entitatea SE\_SERVESTE\_CU are urmatoarele atribute:

* pizza\_id= variabila de tip caracter, de lungime maxima 10, care reprezinta codul pizzei, atribut care corespunde valorii unei chei primare din PIZZA;
* souce\_id= variabila de tip caracter, de lungime maxima 10, care reprezinta codul sosului cu care se serveste pizza, atribut care corespunde valorii unei chei primare din SOUCES;



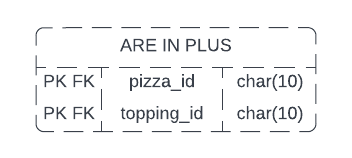
Entitatea SOUCES are urmatoarele atribute:

* souce\_id= variabila de tip caracter, de lungime maxima 10, care reprezinta codul sosului, este cheie primara;
* souce\_type= variabila de tip caracter, de lungime maxima 20, care reprezinta denumirea sosului, NOT NULL;
* souce\_price\_addition= variabila de tip intreg, de lungime maxima 3 cu maxim 2 zecimale, care reprezinta pretul sosului;



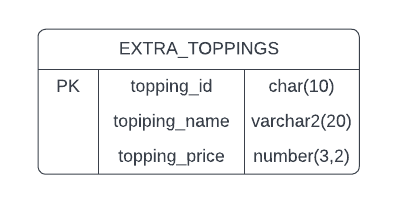
Relatia PIZZA\_are in plus\_EXTRA\_TOPPINGS=> entitatea ARE\_IN\_PLUS are urmatoarele atribute:

* pizza\_id= variabila de tip caracter, de lungime maxima 10, care reprezinta codul pizzei, atribut care corespunde valorii unei chei primare din PIZZA;
* topping\_id= variabila de tip caracter, de lungime maxima 10, care reprezinta codul topingului adaugat, atribut care corespunde valorii unei chei primare din EXTRA\_TOPPINGS;



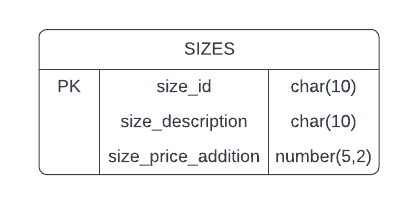
Entitatea EXTRA\_TOPPINGS are urmatoarele atribute:

* topping\_id= variabila de tip caracter, de lungime maxima 10, care reprezinta codul topingului, este cheie primara;
* topping\_name= variabila de tip caracter, de lungime maxima 20, care reprezinta denumirea topingului, NOT NULL;
* topping\_price= variabila de tip intreg, de lungime maxima 3 cu maxim 2 zecimale, care reprezinta pretul topingului, NOT NULL;



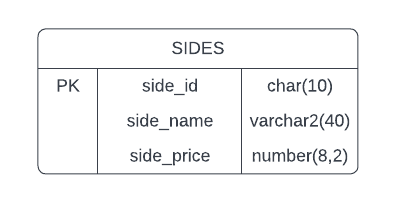
Entitatea SIZES are urmatoarele atribute:

* size\_id= variabila de tip caracter, de lungime maxima 10, care reprezinta codul dimensiunii, este cheie primara;
* size\_description= variabila de tip caracter, de lungime maxima 10, care reprezinta descrierea dimensiunii, NOT NULL;
* size\_price\_addition= variabila de tip intreg, de lungime maxima 5 cu maxim 2 zecimale, care reprezinta pretul pentru dimensiune;



Superentitatea SIDES are urmatoarele atribute:

* side\_id= variabila de tip caracter, de lungime maxima 10, care reprezinta codul alimentului, este cheie primara;
* side\_name= variabila de tip caracter, de lungime maxima 40, care reprezinta denumirea alimentului, NOT NULL;
* side\_price= variabila de tip intreg, de lungime maxima 8 cu maxim 2 zecimale, care reprezinta pretul alimentului, NOT NULL;



Subentitatea DRINKS are urmatoarele atribute:

* side\_id= variabila de tip caracter, de lungime maxima 10, care reprezinta codul alimentului, este cheie primara;
* side\_name= variabila de tip caracter, de lungime maxima 40, care reprezinta denumirea alimentului, NOT NULL;
* side\_price= variabila de tip intreg, de lungime maxima 8 cu maxim 2 zecimale, care reprezinta pretul alimentului, NOT NULL;



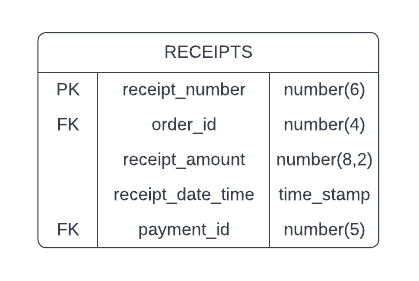
Subentitatea SWEETS are urmatoarele atribute:

* side\_id= variabila de tip caracter, de lungime maxima 10, care reprezinta codul alimentului, este cheie primara;
* side\_name= variabila de tip caracter, de lungime maxima 40, care reprezinta denumirea alimentului, NOT NULL;
* side\_price= variabila de tip intreg, de lungime maxima 8 cu maxim 2 zecimale, care reprezinta pretul alimentului, NOT NULL;



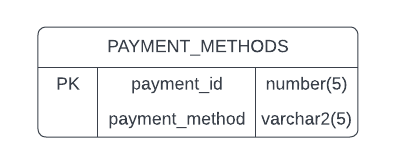
Entitatea RECEIPTS are urmatoarele atribute:

* receipt\_number= variabila de tip intreg, de lungime 6, care reprezinta numarul notei de plata, este cheie primara;
* order\_id= variabila de tip intreg, de lungime 4, care reprezinta codul comenzii, este cheie secundara;
* receipt\_amount= variabila de tip intreg, de lungime maxima 8 cu maxim 2 zecimale, care reprezinta pretul comenzii, NOT NULL;
* receipt\_date\_time= variabila de tip timestamp, care reprezinta data si timpul la care a fost emisa nota de plata, NOT NULL;
* payment\_id= variabila de tip intreg, de lungime 5, care reprezinta codul platii efectuate pentru comanda, este cheie secundar;

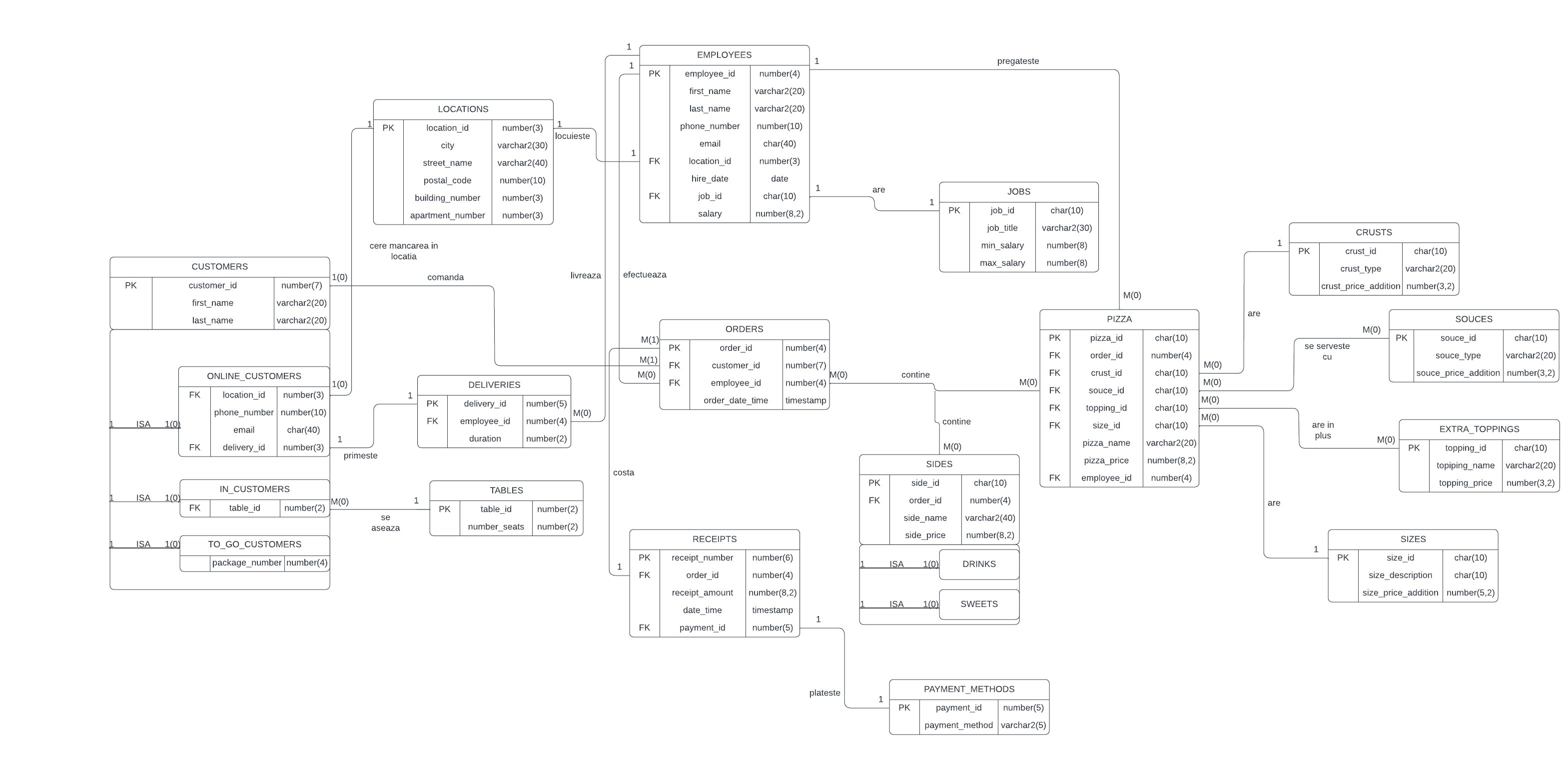


Entitatea PAYMENT\_METHODS are urmatoarele atribute:

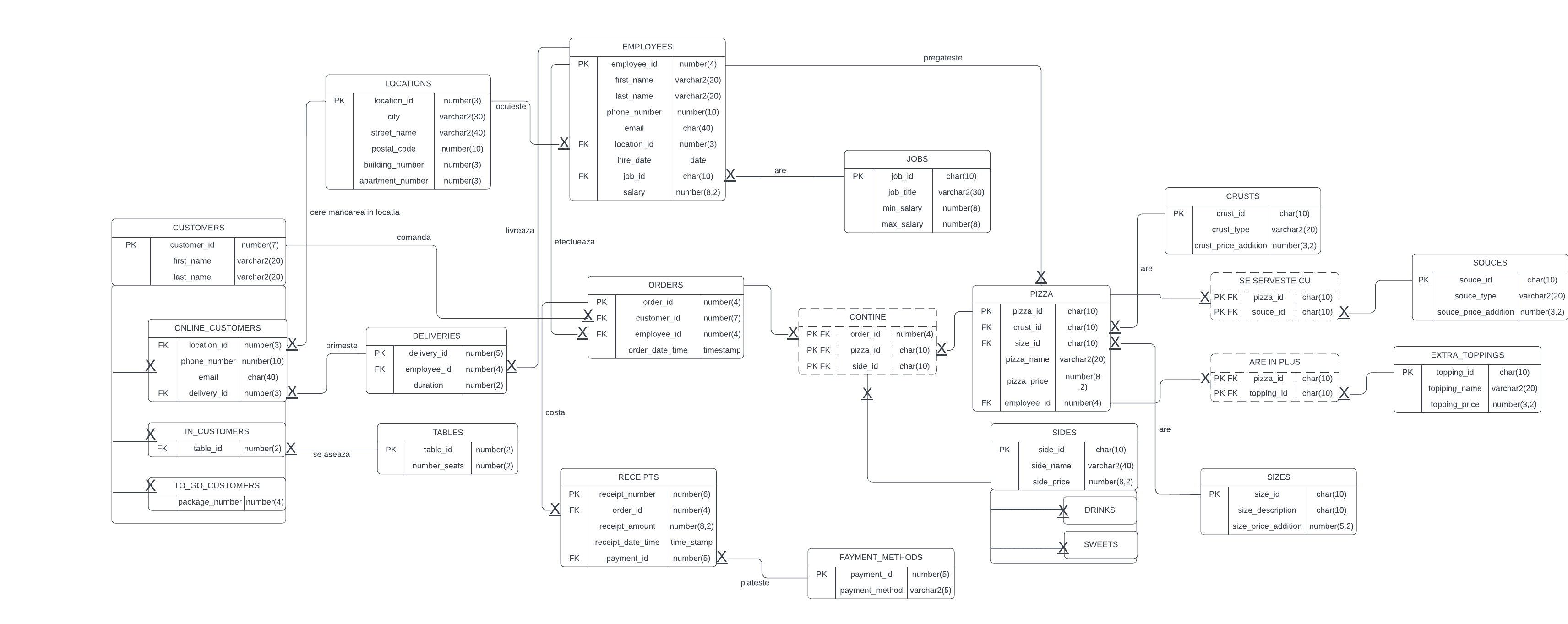
* payment\_id= variabila de tip intreg, de lungime 5, care reprezinta codul platii, este cheie primara;
* payment\_method= variabila de tip intreg, de lungime maxima 5, care reprezinta tipul de plata, cash sau card, NOT NULL;



EX6. Realizarea diagramei entitate-relație corespunzătoare descrierii de la punctele 3-5.



EX7. Realizarea diagramei conceptuale corespunzătoare diagramei entitate-relație proiectate la punctul 6. Diagrama conceptuală obținută trebuie să conțină minimum 6 tabele (fără considerarea subentităților), dintre care cel puțin un tabel asociativ.



EX8. Enumerarea schemelor relaționale corespunzătoare diagramei conceptuale proiectate la punctul 7.

CUSTOMERS (customer\_id#, first\_name, last\_name);

ONLINE\_CUSTOMERS (customer\_id#, first\_name, last\_name, location\_id#, phone\_number, email, delivery\_id#);

IN\_CUSTOMERS (customer\_id#, first\_name, last\_name, table\_id#);

TO\_GO\_CUSTOMERS (customer\_id#, first\_name, last\_name, package\_number);

LOCATIONS (location\_id#, city, street\_name, postal\_code, building\_number, apartment\_number);

DELIVERIES (delivery\_id#, employee\_id#, duration);

TABLES (table\_id#, number\_seats);

EMPLOYEES (employee\_id#, first\_name, last\_name, phone\_number, email, location\_id#, hire\_date, job\_id#, salary);

JOBS (job\_id#, job\_title, min\_salary, max\_salary);

ORDERS (order\_id#, customer\_id#, employee\_id#, order\_date\_time);

CONTINE (order\_id#, pizza\_id#, side\_id#);

PIZZA (pizza\_id#, crust\_id#, size\_id#, pizza\_name, pizza\_price, employee\_id#);

CRUSTS (crust\_id#, crust\_type, crust\_price\_addition);

SE\_SERVESTE\_CU (pizza\_id#, souce\_id#);

SOUCES (souce\_id#, souce\_type, souce\_price\_addition);

ARE\_IN\_PLUS (pizza\_id#, topping\_id#);

EXTRA\_TOPPINGS (topping\_id#, topping\_name, topping\_price);

SIZES (size\_id#, size\_description, size\_price\_addition);

SIDES (side\_id#, side\_name, side\_price);

DRINKS (side\_id#, side\_name, side\_price);

SWEETS (side\_id#, side\_name, side\_price);

RECEIPTS (receipt\_number#, order\_id#, receipt\_amount, receipt\_date\_time, payment\_id#);

PAYMENT\_METHODS (payment\_id#, payment\_method);

EX9. Realizarea normalizării până la forma normală 3(FN1-FN3).

O relaţie este în **FN1** dacă domeniile pe care sunt definite atributele relaţiei sunt constituite numai din valori atomice. Un tuplu nu trebuie să conţină atribute sau grupuri de atribute repetitive.

Aducerea relaţiilor în FN1 presupune eliminarea atributelor compuse şi a celor repetitive.

Voi ilustra printr-un exemplu folosind entitaile SIDES si ORDERS faptul ca baza de data se afla in forma normala FN1. SIDES este o superentitate care are ca subunitati DRINKS si SWEETS pentru a nu avea mai multe valori semnificative in acelasi camp.

|  |  |
| --- | --- |
| ORDERS | SIDES |
| 301 | Coca Cola, Inghetata de vanilie |
| 302 | Cappuccino, Clatite cu ciocolata |
| 303 | Irish Coffee |

|  |  |  |
| --- | --- | --- |
| ORDERS | SIDES | |
| DRINKS | SWEETS |
| 301 | Coca Cola | Inghetata de vanilie |
| 302 | Cappuccino | Clatite cu ciocolata |
| 303 | Irish Coffee | - |

Voi ilustra un alt exemplu utilizand entitatile ORDERS si PIZZA pentru a demonstra faptul ca baza de date nu are mai multe coloane care reprezinta acelasi tip de date/fapte/obiecte.

|  |  |  |  |
| --- | --- | --- | --- |
| ORDERS | PIZZA(1) | PIZZA(2) | PIZZA(3) |
| 301 | Capriciosa | Quatro\_Formaggi | Prosciutto\_Funghi |
| 302 | Quatro\_Fomaggi | Prosciutto\_Funghi | Pollo |
| 303 | Diavola |  |  |

|  |  |
| --- | --- |
| ORDERS | PIZZA |
| 301 | Capriciosa |
| 302 | Quatro\_Formaggi |
| 303 | Diavola |
| 301 | Quatro\_Formaggi |
| 302 | Prosciutto\_Funghi |
| 301 | Prosciutto\_Funghi |
| 302 | Pollo |

|  |  |
| --- | --- |
| ORDERS | PIZZA |
| 301 | Capriciosa |
| 302 | Quatro\_Formaggi |
| 301 | Quatro\_Formaggi |
| 302 | Prosciutto\_Funghi |
| 301 | Prosciutto\_Funghi |
| 303 | Diavola |
| 302 | Pollo |

|  |  |  |
| --- | --- | --- |
| ID | ORDERS | PIZZA |
| 1 | 301 | Capriciosa |
| 2 | 302 | Quatro\_Formaggi |
| 3 | 301 | Quatro\_Formaggi |
| 4 | 302 | Prosciutto\_Funghi |
| 5 | 301 | Prosciutto\_Funghi |
| 6 | 303 | Diavola |
| 7 | 302 | Pollo |

**FN2** cere ca toate elementele unei tabele sa fie dependente functional de totalitatea cheii primare.

Daca unul sau mai multe elemente sunt dependente functional numai de o parte a cheii primare, atunci ele trebuie sa fie separate in tabele diferite.

Daca tabela are o cheie primara formata din numai un atribut, atunci ea este automat in **FN2**.

Voi ilustra un exemplu prin entitatea EMPLOYEES.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Employee\_id | First\_name | Last\_name | Salary | Job\_id | Job\_title |
| 103 | Raluca | Ionescu | 20000 | W\_1 | Chelner |
| 104 | Matei | Popescu | 40000 | CH\_1 | Bucatar |
| 105 | Carina | Maftei | 60000 | MG | Manager |

Cheia primara este o cheie compusa, formata din Employee\_id si Job\_id.

Job\_title depinde numai de Job\_id nu si de Employee\_id.

In FN2 tabelul este urmatorul:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Employee\_id | First\_name | Last\_name | Salary | Job\_id | Job\_title |
| 103 | Raluca | Ionescu | 20000 | W\_1 | Chelner |
| 104 | Matei | Popescu | 40000 | CH\_1 | Bucatar |
| 105 | Carina | Maftei | 60000 | MG | Manager |

|  |  |  |  |
| --- | --- | --- | --- |
| Employee\_id | First\_name | Last\_name | Salary |
| 103 | Raluca | Ionescu | 20000 |
| 104 | Matei | Popescu | 40000 |
| 105 | Carina | Maftei | 60000 |

|  |  |
| --- | --- |
| Job\_id | Job\_title |
| W\_1 | Chelner |
| CH\_1 | Bucatar |
| MG | Manager |

O relaţie este în forma normală trei **FN3** dacă:

1. se găseşte în FN2

2. fiecare atribut care nu este cheie (nu participă la o cheie) depinde direct de cheia primară.

A treia regulă de normalizare cere ca toate câmpurile din tabele să fie independente între ele.

Voi ilustra un exemplu utilizand entitatea PIZZA.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Pizza\_id | Crust\_type | Crust\_price | Size\_description | Size\_price | Pizza\_name | Pizza\_price |
| QF | Pufos | 8 | 6 felii | 15 | Quatro\_Formaggi | 42 |
| DIV | Normal | 0 | 8 felii | 10 | Diavola | 35 |

In exemplu se poate observa faptul ca pizza depinde de blat si de dimensiune.

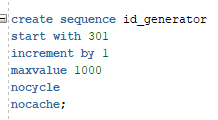
Pentru a aduce tabela in FN3, separt atributele legate de blat si cele legat de dimensiune.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pizza\_id | Crust\_id | Size\_id | Pizza\_name | Pizza\_price |
| QF | PUF | MICA | Quatro\_Formaggi | 42 |
| DIV | NOR | MEDIE | Diavola | 35 |

|  |  |  |
| --- | --- | --- |
| Crust\_id | Crust\_type | Crust\_price |
| PUF | Pufos | 8 |
| NOR | Normal | 0 |

|  |  |  |
| --- | --- | --- |
| Size\_id | Size\_description | Size\_price |
| MICA | 6 felii | 15 |
| MEDIE | 8 felii | 10 |

EX10. Crearea unei secvențe ce va fi utilizată în inserarea înregistrărilor în tabele (punctul 11).



create sequence id\_generator

start with 301

increment by 1

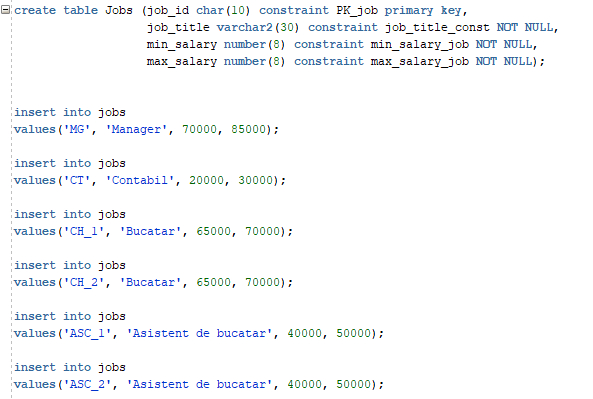
maxvalue 1000

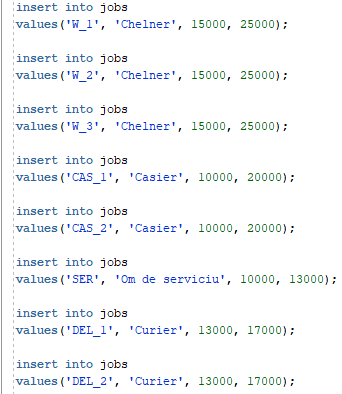
nocycle

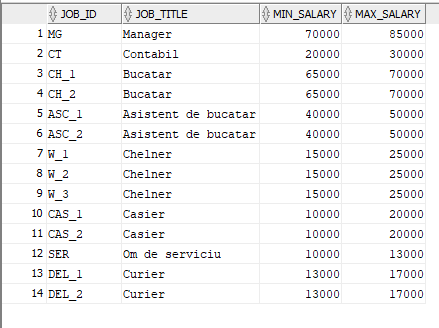
nocache;

EX11. Crearea tabelelor în SQL și inserarea de date coerente în fiecare dintre acestea (minimum 5 înregistrări în fiecare table neasociativ; minimum 10 înregistrări în tabelele asociative).

**JOBS**







create table Jobs (job\_id char(10) constraint PK\_job primary key,

job\_title varchar2(30) constraint job\_title\_const NOT NULL,

min\_salary number(8) constraint min\_salary\_job NOT NULL,

max\_salary number(8) constraint max\_salary\_job NOT NULL);

insert into jobs

values('MG', 'Manager', 70000, 85000);

insert into jobs

values('CT', 'Contabil', 20000, 30000);

insert into jobs

values('CH\_1', 'Bucatar', 65000, 70000);

insert into jobs

values('CH\_2', 'Bucatar', 65000, 70000);

insert into jobs

values('ASC\_1', 'Asistent de bucatar', 40000, 50000);

insert into jobs

values('ASC\_2', 'Asistent de bucatar', 40000, 50000);

insert into jobs

values('W\_1', 'Chelner', 15000, 25000);

insert into jobs

values('W\_2', 'Chelner', 15000, 25000);

insert into jobs

values('W\_3', 'Chelner', 15000, 25000);

insert into jobs

values('CAS\_1', 'Casier', 10000, 20000);

insert into jobs

values('CAS\_2', 'Casier', 10000, 20000);

insert into jobs

values('SER', 'Om de serviciu', 10000, 13000);

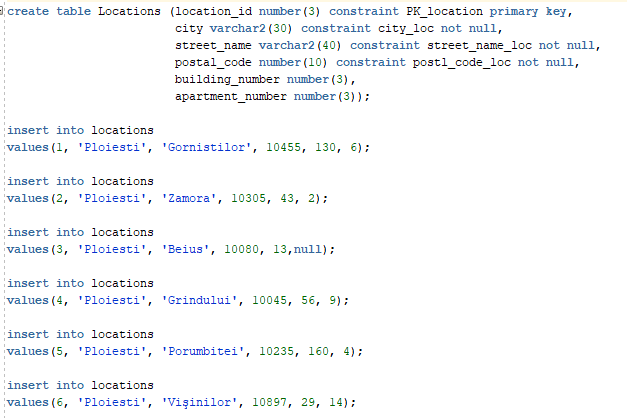
insert into jobs

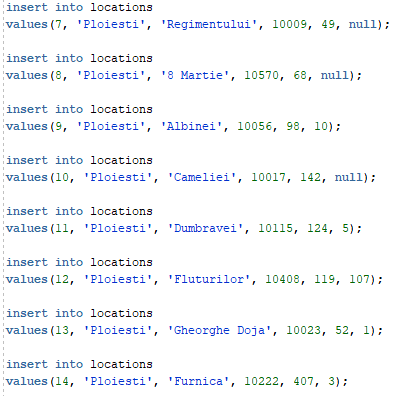
values('DEL\_1', 'Curier', 13000, 17000);

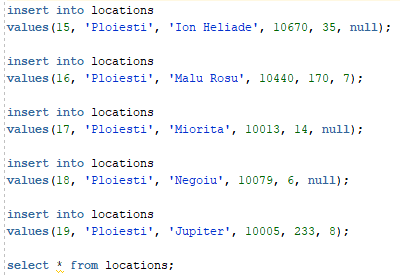
insert into jobs

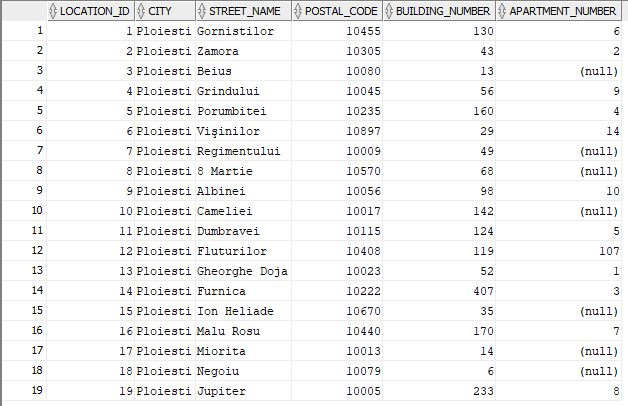
values('DEL\_2', 'Curier', 13000, 17000);

**LOCATIONS**









create table Locations (location\_id number(3) constraint PK\_location primary key,

city varchar2(30) constraint city\_loc not null,

street\_name varchar2(40) constraint street\_name\_loc not null,

postal\_code number(10) constraint postl\_code\_loc not null,

building\_number number(3),

apartment\_number number(3));

insert into locations

values(1, 'Ploiesti', 'Gornistilor', 10455, 130, 6);

insert into locations

values(2, 'Ploiesti', 'Zamora', 10305, 43, 2);

insert into locations

values(3, 'Ploiesti', 'Beius', 10080, 13,null);

insert into locations

values(4, 'Ploiesti', 'Grindului', 10045, 56, 9);

insert into locations

values(5, 'Ploiesti', 'Porumbitei', 10235, 160, 4);

insert into locations

values(6, 'Ploiesti', 'Vi?inilor', 10897, 29, 14);

insert into locations

values(7, 'Ploiesti', 'Regimentului', 10009, 49, null);

insert into locations

values(8, 'Ploiesti', '8 Martie', 10570, 68, null);

insert into locations

values(9, 'Ploiesti', 'Albinei', 10056, 98, 10);

insert into locations

values(10, 'Ploiesti', 'Cameliei', 10017, 142, null);

insert into locations

values(11, 'Ploiesti', 'Dumbravei', 10115, 124, 5);

insert into locations

values(12, 'Ploiesti', 'Fluturilor', 10408, 119, 107);

insert into locations

values(13, 'Ploiesti', 'Gheorghe Doja', 10023, 52, 1);

insert into locations

values(14, 'Ploiesti', 'Furnica', 10222, 407, 3);

insert into locations

values(15, 'Ploiesti', 'Ion Heliade', 10670, 35, null);

insert into locations

values(16, 'Ploiesti', 'Malu Rosu', 10440, 170, 7);

insert into locations

values(17, 'Ploiesti', 'Miorita', 10013, 14, null);

insert into locations

values(18, 'Ploiesti', 'Negoiu', 10079, 6, null);

insert into locations

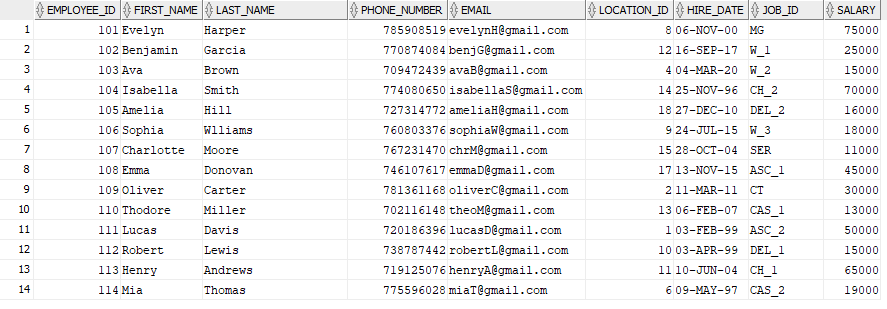
values(19, 'Ploiesti', 'Jupiter', 10005, 233, 8);

**EMPLOYEES**









create table Employees (employee\_id number(4) constraint PK\_employee primary key,

first\_name varchar2(20) constraint last\_name\_emp not null,

last\_name varchar2(20) constraint first\_name\_emp not null,

phone\_number number(10) unique,

email varchar2(40) unique,

location\_id number(30) constraint FK\_emp\_1 references Locations(location\_id),

hire\_date date constraint hire\_date\_emp not null,

job\_id char(10) constraint FK\_emp\_2 references Jobs (job\_id),

salary number(8,2) constraint salary\_emp not null);

insert into employees

values(101, 'Evelyn', 'Harper', 0785908519,'evelynH@gmail.com', 8, '06-NOV-00', 'MG', 75000);

insert into employees

values(102, 'Benjamin', 'Garcia', 0770874084,'benjG@gmail.com', 12, '16-SEP-17', 'W\_1', 25000);

insert into employees

values(103, 'Ava', 'Brown', 0709472439,'avaB@gmail.com', 4, '04-MAR\_20', 'W\_2', 15000);

insert into employees

values(104, 'Isabella', 'Smith', 0774080650,'isabellaS@gmail.com', 14, '25-NOV-96', 'CH\_2', 70000);

insert into employees

values(105, 'Amelia', 'Hill', 0727314772,'ameliaH@gmail.com', 18, '27-DEC-10', 'DEL\_2', 16000);

insert into employees

values(106, 'Sophia', 'Wlliams', 0760803376,'sophiaW@gmail.com', 9, '24-JUL-15', 'W\_3', 18000);

insert into employees

values(107, 'Charlotte', 'Moore', 0767231470,'chrM@gmail.com', 15, '28-OCT\_04', 'SER', 11000);

insert into employees

values(108, 'Emma', 'Donovan', 0746107617,'emmaD@gmail.com', 17, '13-NOV-15', 'ASC\_1', 45000);

insert into employees

values(109, 'Oliver', 'Carter', 0781361168,'oliverC@gmail.com', 2, '11\_MAR-11', 'CT', 30000);

insert into employees

values(110, 'Thodore', 'Miller', 0702116148,'theoM@gmail.com', 13, '06-FEB-07', 'CAS\_1', 13000);

insert into employees

values(111, 'Lucas', 'Davis', 0720186396,'lucasD@gmail.com', 1, '03-FEB-99', 'ASC\_2', 50000);

insert into employees

values(112, 'Robert', 'Lewis', 0738787442,'robertL@gmail.com', 10, '03-APR-99', 'DEL\_1', 15000);

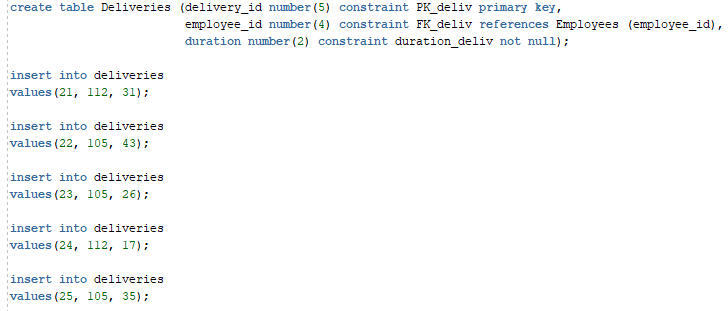
insert into employees

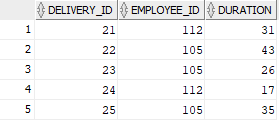
values(113, 'Henry', 'Andrews', 0719125076,'henryA@gmail.com', 11, '10-JUN-04', 'CH\_1', 65000);

insert into employees

values(114, 'Mia', 'Thomas', 0775596028,'miaT@gmail.com', 6, '09-MAY-97', 'CAS\_2', 19000);

**DELIVERIES**





create table Deliveries (delivery\_id number(5) constraint PK\_deliv primary key,

employee\_id number(4) constraint FK\_deliv references Employees (employee\_id),

duration number(2) constraint duration\_deliv not null);

insert into deliveries

values(21, 112, 31);

insert into deliveries

values(22, 105, 43);

insert into deliveries

values(23, 105, 26);

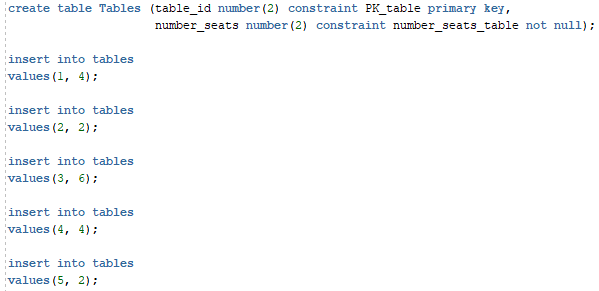
insert into deliveries

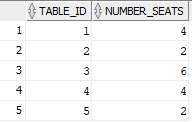
values(24, 112, 17);

insert into deliveries

values(25, 105, 35);

**TABLES**





create table Tables (table\_id number(2) constraint PK\_table primary key,

number\_seats number(2) constraint number\_seats\_table not null);

insert into tables

values(1, 4);

insert into tables

values(2, 2);

insert into tables

values(3, 6);

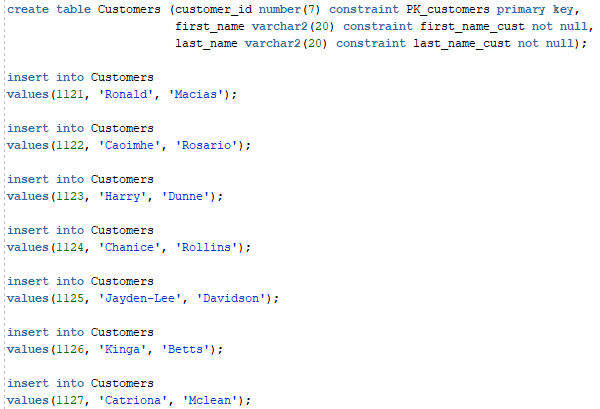
insert into tables

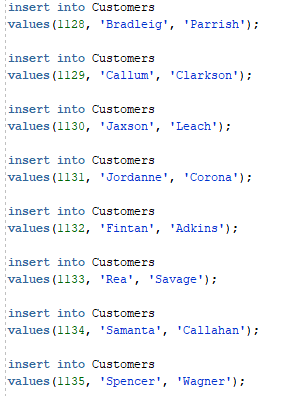
values(4, 4);

insert into tables

values(5, 2);

**CUSTOMERS**









create table Customers (customer\_id number(7) constraint PK\_customers primary key,

first\_name varchar2(20) constraint first\_name\_cust not null,

last\_name varchar2(20) constraint last\_name\_cust not null);

insert into Customers

values(1121, 'Ronald', 'Macias');

insert into Customers

values(1122, 'Caoimhe', 'Rosario');

insert into Customers

values(1123, 'Harry', 'Dunne');

insert into Customers

values(1124, 'Chanice', 'Rollins');

insert into Customers

values(1125, 'Jayden-Lee', 'Davidson');

insert into Customers

values(1126, 'Kinga', 'Betts');

insert into Customers

values(1127, 'Catriona', 'Mclean');

insert into Customers

values(1128, 'Bradleig', 'Parrish');

insert into Customers

values(1129, 'Callum', 'Clarkson');

insert into Customers

values(1130, 'Jaxson', 'Leach');

insert into Customers

values(1131, 'Jordanne', 'Corona');

insert into Customers

values(1132, 'Fintan', 'Adkins');

insert into Customers

values(1133, 'Rea', 'Savage');

insert into Customers

values(1134, 'Samanta', 'Callahan');

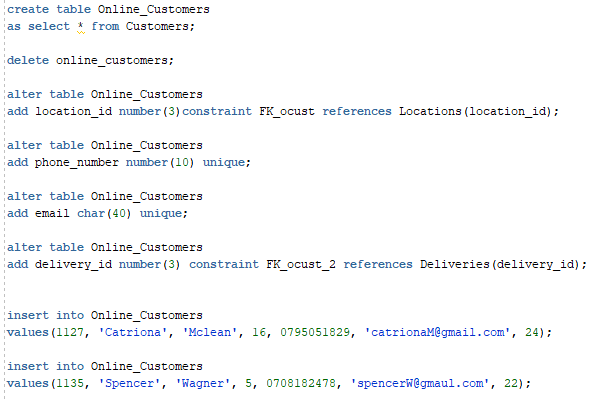
insert into Customers

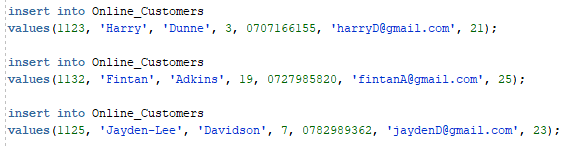
values(1135, 'Spencer', 'Wagner');

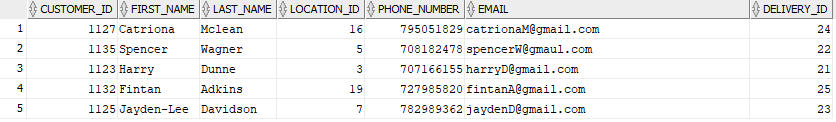
insert into Customers

values(1136, 'Karen', 'Willow');

**ONLINE\_CUSTOMERS**







create table Online\_Customers

as select \* from Customers;

delete online\_customers;

alter table Online\_Customers

add location\_id number(3)constraint FK\_ocust references Locations(location\_id);

alter table Online\_Customers

add phone\_number number(10) unique;

alter table Online\_Customers

add email char(40) unique;

alter table Online\_Customers

add delivery\_id number(3) constraint FK\_ocust\_2 references Deliveries(delivery\_id);

insert into Online\_Customers

values(1127, 'Catriona', 'Mclean', 16, 0795051829, 'catrionaM@gmail.com', 24);

insert into Online\_Customers

values(1135, 'Spencer', 'Wagner', 5, 0708182478, 'spencerW@gmaul.com', 22);

insert into Online\_Customers

values(1123, 'Harry', 'Dunne', 3, 0707166155, 'harryD@gmail.com', 21);

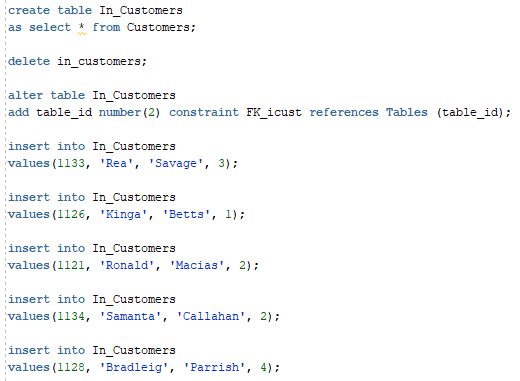
insert into Online\_Customers

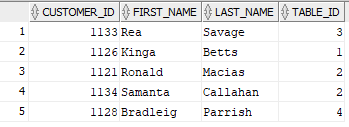
values(1132, 'Fintan', 'Adkins', 19, 0727985820, 'fintanA@gmail.com', 25);

insert into Online\_Customers

values(1125, 'Jayden-Lee', 'Davidson', 7, 0782989362, 'jaydenD@gmail.com', 23);

**IN\_CUSTOMERS**





create table In\_Customers

as select \* from Customers;

delete in\_customers;

alter table In\_Customers

add table\_id number(2) constraint FK\_icust references Tables (table\_id);

insert into In\_Customers

values(1133, 'Rea', 'Savage', 3);

insert into In\_Customers

values(1126, 'Kinga', 'Betts', 1);

insert into In\_Customers

values(1121, 'Ronald', 'Macias', 2);

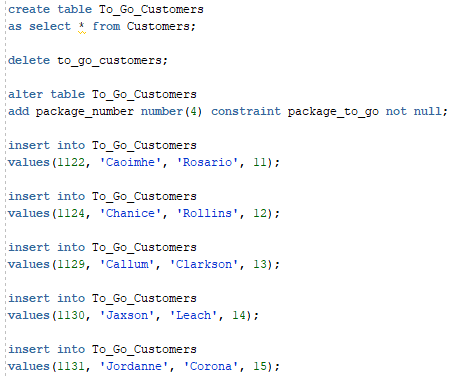
insert into In\_Customers

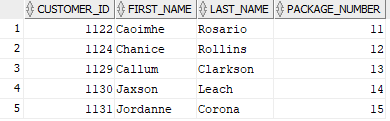
values(1134, 'Samanta', 'Callahan', 2);

insert into In\_Customers

values(1128, 'Bradleig', 'Parrish', 4);

**TO\_GO\_CUSTOMERS**





create table To\_Go\_Customers

as select \* from Customers;

delete to\_go\_customers;

alter table To\_Go\_Customers

add package\_number number(4) constraint package\_to\_go not null;

insert into To\_Go\_Customers

values(1122, 'Caoimhe', 'Rosario', 11);

insert into To\_Go\_Customers

values(1124, 'Chanice', 'Rollins', 12);

insert into To\_Go\_Customers

values(1129, 'Callum', 'Clarkson', 13);

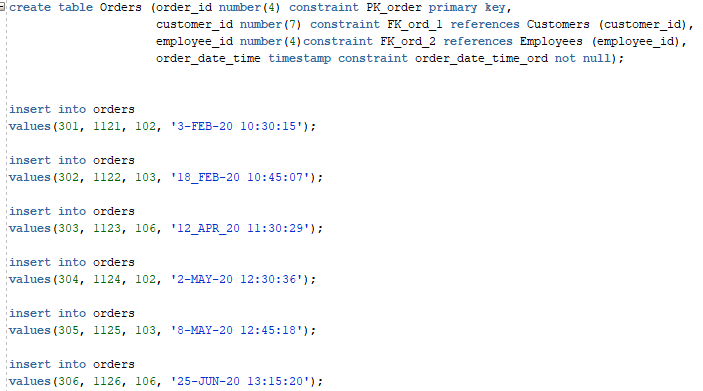
insert into To\_Go\_Customers

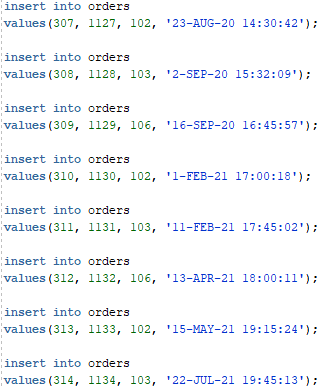
values(1130, 'Jaxson', 'Leach', 14);

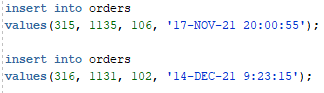
insert into To\_Go\_Customers

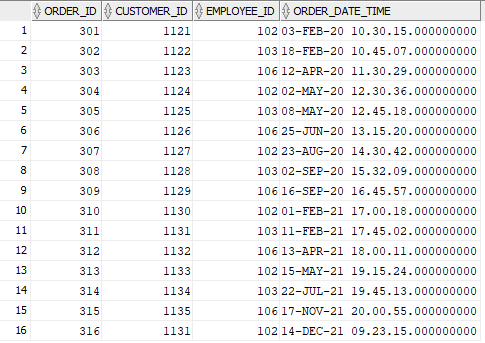
values(1131, 'Jordanne', 'Corona', 15);

**ORDERS**









create table Orders (order\_id number(4) constraint PK\_order primary key,

customer\_id number(7) constraint FK\_ord\_1 references Customers (customer\_id),

employee\_id number(4)constraint FK\_ord\_2 references Employees (employee\_id),

order\_date\_time timestamp constraint order\_date\_time\_ord not null);

insert into orders

values(301, 1121, 102, '3-FEB-20 10:30:15');

insert into orders

values(302, 1122, 103, '18\_FEB-20 10:45:07');

insert into orders

values(303, 1123, 106, '12\_APR\_20 11:30:29');

insert into orders

values(304, 1124, 102, '2-MAY-20 12:30:36');

insert into orders

values(305, 1125, 103, '8-MAY-20 12:45:18');

insert into orders

values(306, 1126, 106, '25-JUN-20 13:15:20');

insert into orders

values(307, 1127, 102, '23-AUG-20 14:30:42');

insert into orders

values(308, 1128, 103, '2-SEP-20 15:32:09');

insert into orders

values(309, 1129, 106, '16-SEP-20 16:45:57');

insert into orders

values(310, 1130, 102, '1-FEB-21 17:00:18');

insert into orders

values(311, 1131, 103, '11-FEB-21 17:45:02');

insert into orders

values(312, 1132, 106, '13-APR-21 18:00:11');

insert into orders

values(313, 1133, 102, '15-MAY-21 19:15:24');

insert into orders

values(314, 1134, 103, '22-JUL-21 19:45:13');

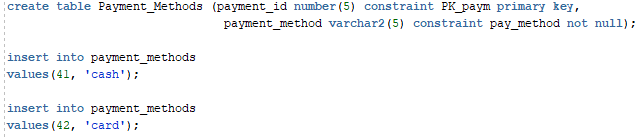
insert into orders

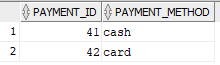
values(315, 1135, 106, '17-NOV-21 20:00:55');

insert into orders

values(316, 1131, 102, '14-DEC-21 9:23:15');

**PAYMENT\_METHODS**





create table Payment\_Methods (payment\_id number(5) constraint PK\_paym primary key,

payment\_method varchar2(5) constraint pay\_method not null);

insert into payment\_methods

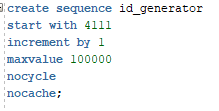
values(41, 'cash');

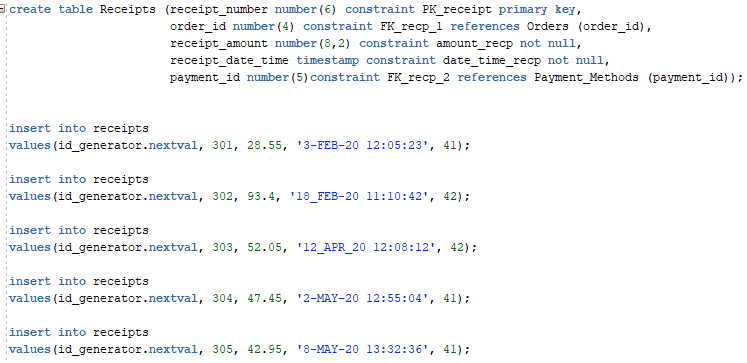
insert into payment\_methods

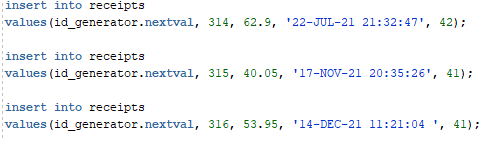
values(42, 'card');

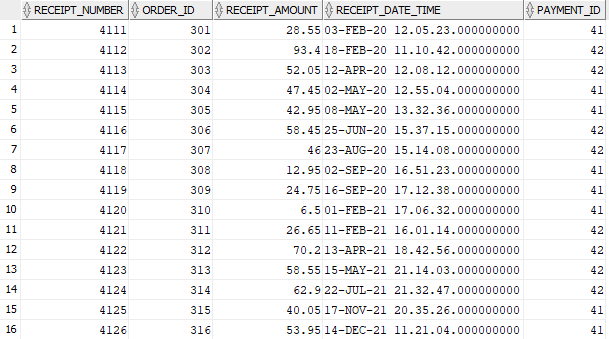
select \* from payment\_methods;

**RECEIPTS**







create sequence id\_generator

start with 4111

increment by 1

maxvalue 100000

nocycle

nocache;

create table Receipts (receipt\_number number(6) constraint PK\_receipt primary key,

order\_id number(4) constraint FK\_recp\_1 references Orders (order\_id),

receipt\_amount number(8,2) constraint amount\_recp not null,

receipt\_date\_time timestamp constraint date\_time\_recp not null,

payment\_id number(5)constraint FK\_recp\_2 references Payment\_Methods (payment\_id));

insert into receipts

values(id\_generator.nextval, 301, 28.55, '3-FEB-20 12:05:23', 41);

insert into receipts

values(id\_generator.nextval, 302, 93.4, '18\_FEB-20 11:10:42', 42);

insert into receipts

values(id\_generator.nextval, 303, 52.05, '12\_APR\_20 12:08:12', 42);

insert into receipts

values(id\_generator.nextval, 304, 47.45, '2-MAY-20 12:55:04', 41);

insert into receipts

values(id\_generator.nextval, 305, 42.95, '8-MAY-20 13:32:36', 41);

insert into receipts

values(id\_generator.nextval, 306, 58.45, '25-JUN-20 15:37:15', 42);

insert into receipts

values(id\_generator.nextval, 307, 46, '23-AUG-20 15:14:08', 42);

insert into receipts

values(id\_generator.nextval, 308, 12.95, '2-SEP-20 16:51:23', 41);

insert into receipts

values(id\_generator.nextval, 309, 24.75, '16-SEP-20 17:12:38', 41);

insert into receipts

values(id\_generator.nextval, 310, 6.5, '1-FEB-21 17:06:32', 41);

insert into receipts

values(id\_generator.nextval, 311, 26.65, '11-FEB-21 16:01:14 ', 42);

insert into receipts

values(id\_generator.nextval, 312, 70.2, '13-APR-21 18:42:56', 42);

insert into receipts

values(id\_generator.nextval, 313, 58.55, '15-MAY-21 21:14:03', 42);

insert into receipts

values(id\_generator.nextval, 314, 62.9, '22-JUL-21 21:32:47', 42);

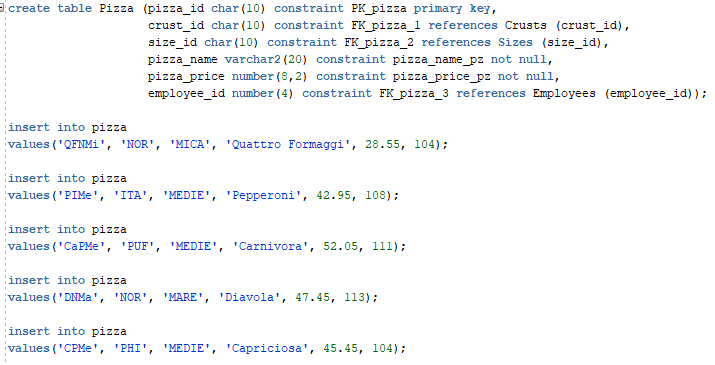
insert into receipts

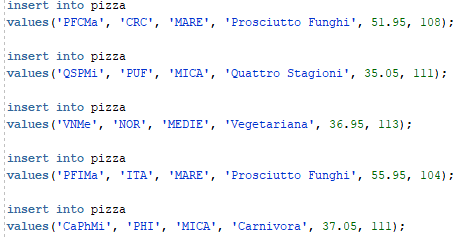
values(id\_generator.nextval, 315, 40.05, '17-NOV-21 20:35:26', 41);

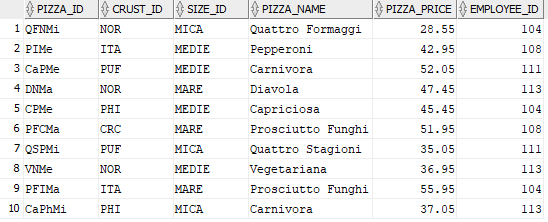
insert into receipts

values(id\_generator.nextval, 316, 53.95, '14-DEC-21 11:21:04 ', 41);

**PIZZA**







create table Pizza (pizza\_id char(10) constraint PK\_pizza primary key,

crust\_id char(10) constraint FK\_pizza\_1 references Crusts (crust\_id),

size\_id char(10) constraint FK\_pizza\_2 references Sizes (size\_id),

pizza\_name varchar2(20) constraint pizza\_name\_pz not null,

pizza\_price number(8,2) constraint pizza\_price\_pz not null,

employee\_id number(4) constraint FK\_pizza\_3 references Employees (employee\_id));

insert into pizza

values('QFNMi', 'NOR', 'MICA', 'Quattro Formaggi', 28.55, 104);

insert into pizza

values('PIMe', 'ITA', 'MEDIE', 'Pepperoni', 42.95, 108);

insert into pizza

values('CaPMe', 'PUF', 'MEDIE', 'Carnivora', 52.05, 111);

insert into pizza

values('DNMa', 'NOR', 'MARE', 'Diavola', 47.45, 113);

insert into pizza

values('CPMe', 'PHI', 'MEDIE', 'Capriciosa', 45.45, 104);

insert into pizza

values('PFCMa', 'CRC', 'MARE', 'Prosciutto Funghi', 51.95, 108);

insert into pizza

values('QSPMi', 'PUF', 'MICA', 'Quattro Stagioni', 35.05, 111);

insert into pizza

values('VNMe', 'NOR', 'MEDIE', 'Vegetariana', 36.95, 113);

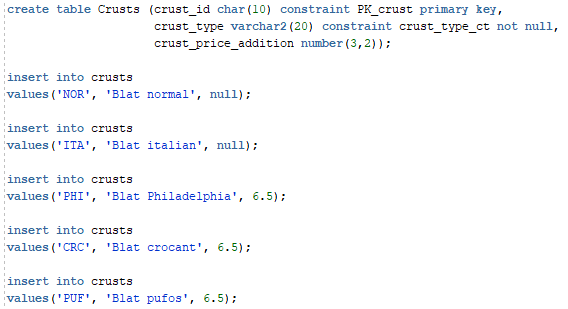
insert into pizza

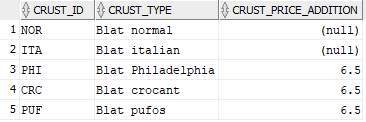
values('PFIMa', 'ITA', 'MARE', 'Prosciutto Funghi', 55.95, 104);

insert into pizza

values('CaPhMi', 'PHI', 'MICA', 'Carnivora', 37.05, 113);

**CRUSTS**





create table Crusts (crust\_id char(10) constraint PK\_crust primary key,

crust\_type varchar2(20) constraint crust\_type\_ct not null,

crust\_price\_addition number(3,2));

insert into crusts

values('NOR', 'Blat normal', null);

insert into crusts

values('ITA', 'Blat italian', null);

insert into crusts

values('PHI', 'Blat Philadelphia', 6.5);

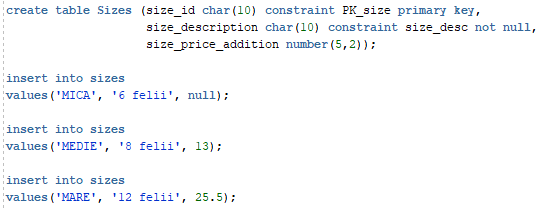
insert into crusts

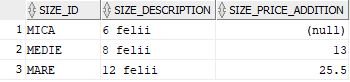
values('CRC', 'Blat crocant', 6.5);

insert into crusts

values('PUF', 'Blat pufos', 6.5);

**SIZES**





create table Sizes (size\_id char(10) constraint PK\_size primary key,

size\_description char(10) constraint size\_desc not null,

size\_price\_addition number(5,2));

insert into sizes

values('MICA', '6 felii', null);

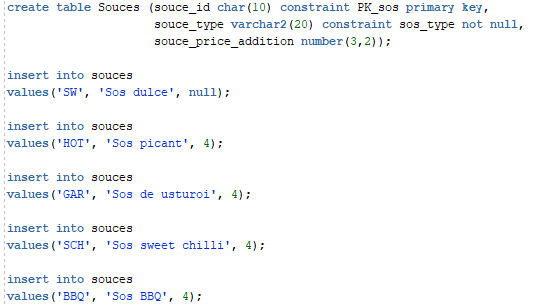
insert into sizes

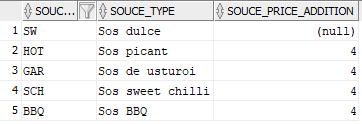
values('MEDIE', '8 felii', 13);

insert into sizes

values('MARE', '12 felii', 25.5);

**SOUCES**





create table Souces (souce\_id char(10) constraint PK\_sos primary key,

souce\_type varchar2(20) constraint sos\_type not null,

souce\_price\_addition number(3,2));

insert into souces

values('SW', 'Sos dulce', null);

insert into souces

values('HOT', 'Sos picant', 4);

insert into souces

values('GAR', 'Sos de usturoi', 4);

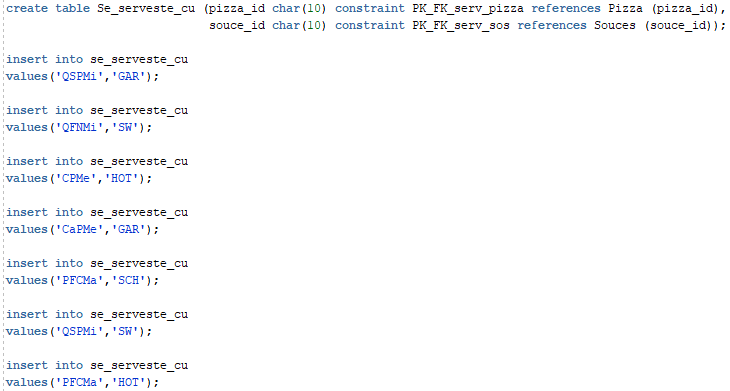
insert into souces

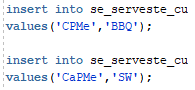
values('SCH', 'Sos sweet chilli', 4);

insert into souces

values('BBQ', 'Sos BBQ', 4);

**SE\_SERVESTE\_CU**







create table Se\_serveste\_cu (pizza\_id char(10) constraint PK\_FK\_serv\_pizza references Pizza (pizza\_id),

souce\_id char(10) constraint PK\_FK\_serv\_sos references Souces (souce\_id));

insert into se\_serveste\_cu

values('QSPMi','GAR');

insert into se\_serveste\_cu

values('QFNMi','SW');

insert into se\_serveste\_cu

values('CPMe','HOT');

insert into se\_serveste\_cu

values('CaPMe','GAR');

insert into se\_serveste\_cu

values('PFCMa','SCH');

insert into se\_serveste\_cu

values('QSPMi','SW');

insert into se\_serveste\_cu

values('PFCMa','HOT');

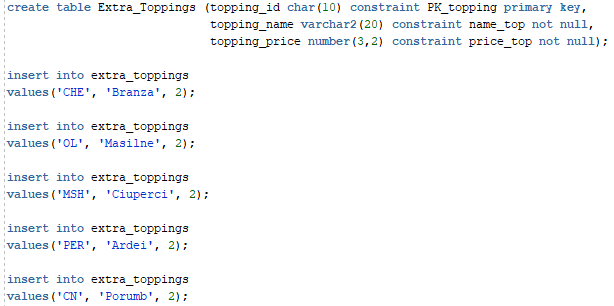
insert into se\_serveste\_cu

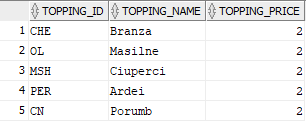
values('CPMe','BBQ');

insert into se\_serveste\_cu

values('CaPMe','SW');

**EXTRA\_TOPPINGS**





create table Extra\_Toppings (topping\_id char(10) constraint PK\_topping primary key,

topping\_name varchar2(20) constraint name\_top not null,

topping\_price number(3,2) constraint price\_top not null);

insert into extra\_toppings

values('CHE', 'Branza', 2);

insert into extra\_toppings

values('OL', 'Masilne', 2);

insert into extra\_toppings

values('MSH', 'Ciuperci', 2);

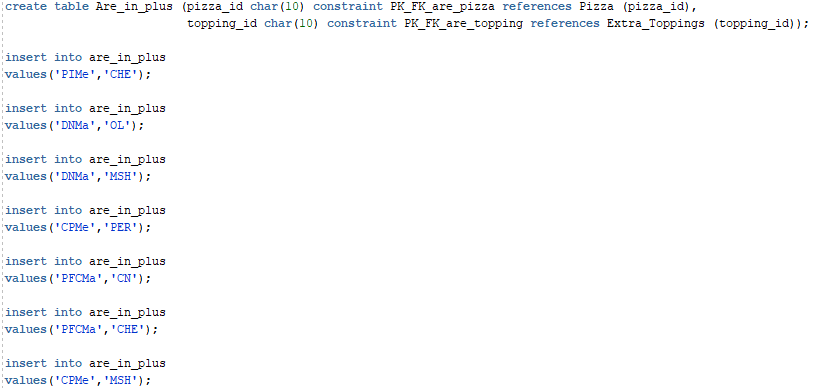
insert into extra\_toppings

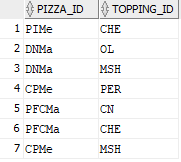
values('PER', 'Ardei', 2);

insert into extra\_toppings

values('CN', 'Porumb', 2);

**ARE\_IN\_PLUS**





create table Are\_in\_plus (pizza\_id char(10) constraint PK\_FK\_are\_pizza references Pizza (pizza\_id),

topping\_id char(10) constraint PK\_FK\_are\_topping references Extra\_Toppings (topping\_id));

insert into are\_in\_plus

values('PIMe','CHE');

insert into are\_in\_plus

values('DNMa','OL');

insert into are\_in\_plus

values('DNMa','MSH');

insert into are\_in\_plus

values('CPMe','PER');

insert into are\_in\_plus

values('PFCMa','CN');

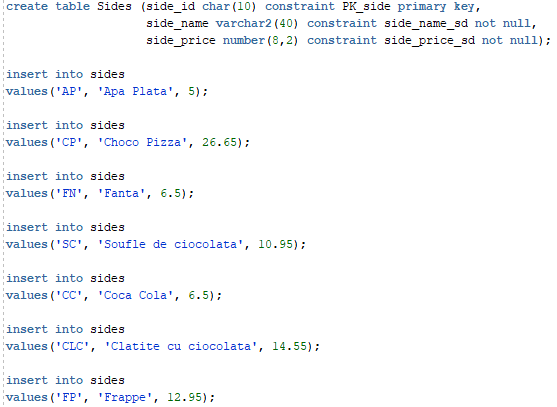
insert into are\_in\_plus

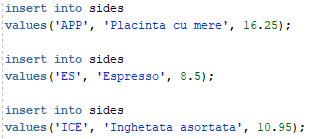
values('PFCMa','CHE');

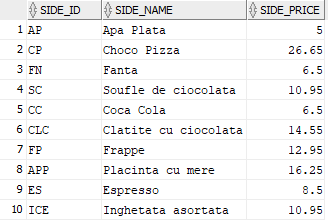
insert into are\_in\_plus

values('CPMe','MSH');

**SIDES**







create table Sides (side\_id char(10) constraint PK\_side primary key,

side\_name varchar2(40) constraint side\_name\_sd not null,

side\_price number(8,2) constraint side\_price\_sd not null);

insert into sides

values('AP', 'Apa Plata', 5);

insert into sides

values('CP', 'Choco Pizza', 26.65);

insert into sides

values('FN', 'Fanta', 6.5);

insert into sides

values('SC', 'Soufle de ciocolata', 10.95);

insert into sides

values('CC', 'Coca Cola', 6.5);

insert into sides

values('CLC', 'Clatite cu ciocolata', 14.55);

insert into sides

values('FP', 'Frappe', 12.95);

insert into sides

values('APP', 'Placinta cu mere', 16.25);

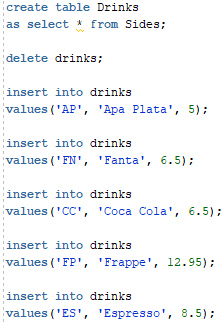
insert into sides

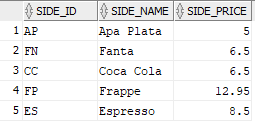
values('ES', 'Espresso', 8.5);

insert into sides

values('ICE', 'Inghetata asortata', 10.95);

**DRINKS**





create table Drinks

as select \* from Sides;

delete drinks;

insert into drinks

values('AP', 'Apa Plata', 5);

insert into drinks

values('FN', 'Fanta', 6.5);

insert into drinks

values('CC', 'Coca Cola', 6.5);

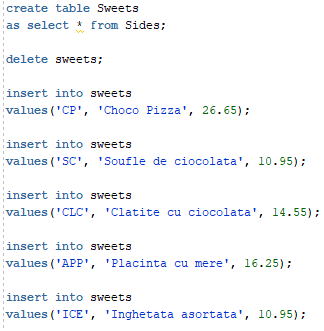
insert into drinks

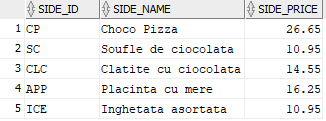
values('FP', 'Frappe', 12.95);

insert into drinks

values('ES', 'Espresso', 8.5);

**SWEETS**





create table Sweets

as select \* from Sides;

delete sweets;

insert into sweets

values('CP', 'Choco Pizza', 26.65);

insert into sweets

values('SC', 'Soufle de ciocolata', 10.95);

insert into sweets

values('CLC', 'Clatite cu ciocolata', 14.55);

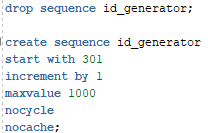
insert into sweets

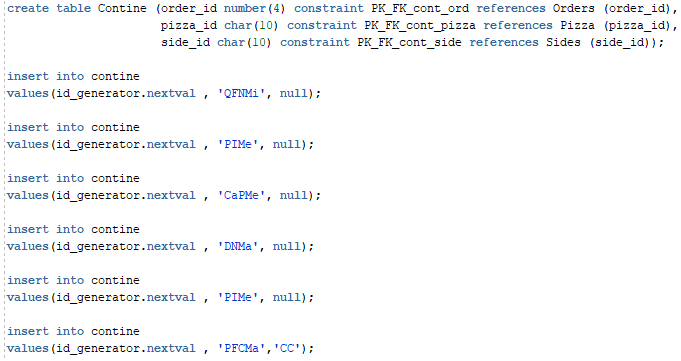
values('APP', 'Placinta cu mere', 16.25);

insert into sweets

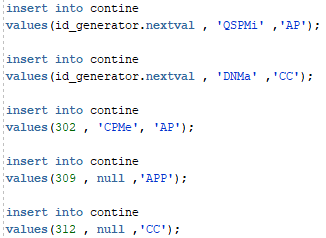
values('ICE', 'Inghetata asortata', 10.95);

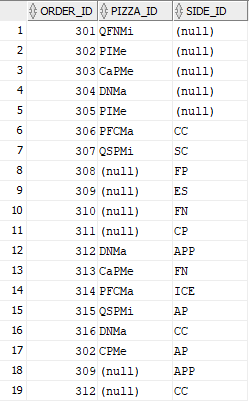
**CONTINE**











drop sequence id\_generator;

create sequence id\_generator

start with 301

increment by 1

maxvalue 1000

nocycle

nocache;

create table Contine (order\_id number(4) constraint PK\_FK\_cont\_ord references Orders (order\_id),

pizza\_id char(10) constraint PK\_FK\_cont\_pizza references Pizza (pizza\_id),

side\_id char(10) constraint PK\_FK\_cont\_side references Sides (side\_id));

insert into contine

values(id\_generator.nextval , 'QFNMi', null);

insert into contine

values(id\_generator.nextval , 'PIMe', null);

insert into contine

values(id\_generator.nextval , 'CaPMe', null);

insert into contine

values(id\_generator.nextval , 'DNMa', null);

insert into contine

values(id\_generator.nextval , 'PIMe', null);

insert into contine

values(id\_generator.nextval , 'PFCMa','CC');

insert into contine

values(id\_generator.nextval , 'QSPMi','SC');

insert into contine

values(id\_generator.nextval , null ,'FP');

insert into contine

values(id\_generator.nextval , null ,'ES');

insert into contine

values(id\_generator.nextval , null ,'FN');

insert into contine

values(id\_generator.nextval , null ,'CP');

insert into contine

values(id\_generator.nextval , 'DNMa' ,'APP');

insert into contine

values(id\_generator.nextval , 'CaPMe' ,'FN');

insert into contine

values(id\_generator.nextval , 'PFCMa','ICE');

insert into contine

values(id\_generator.nextval , 'QSPMi' ,'AP');

insert into contine

values(id\_generator.nextval , 'DNMa' ,'CC');

insert into contine

values(302 , 'CPMe', 'AP');

insert into contine

values(309 , null ,'APP');

insert into contine

values(312 , null ,'CC');

EX12. Formulați în limbaj natural și implementați 5 cereri SQL complexece vor utiliza, în ansamblul lor, următoarele elemente:

•operație join pe cel puțin 4 tabele

•filtrare la nivel de linii

•subcereri sincronizatev în care intervin cel puțin 3 tabele

•subcereri nesincronizatevîn care intervin cel puțin 3 tabele

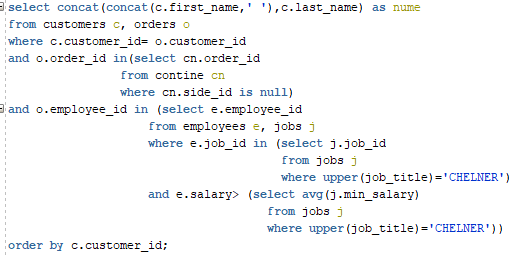
•grupări de date, funcții grup, filtrare la nivel de grupuri

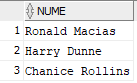
•ordonări

•utilizarea a cel puțin 2 funcții pe șiruri de caractere, 2 funcții pe date calendaristice, a funcțiilor NVL și DECODE, a cel puțin unei expresii CASE

•utilizarea a cel puțin 1 bloc de cerere (clauza WITH)

1. Afisati numele si prenumele clientilor care nu au comandat sides si care au fost ajutati de chelneri cu salariul>min salariului de chelner.





select concat(concat(c.first\_name,' '),c.last\_name) as nume

from customers c, orders o

where c.customer\_id= o.customer\_id

and o.order\_id in(select cn.order\_id

from contine cn

where cn.side\_id is null)

and o.employee\_id in (select e.employee\_id

from employees e, jobs j

where e.job\_id in (select j.job\_id

from jobs j

where upper(job\_title)='CHELNER')

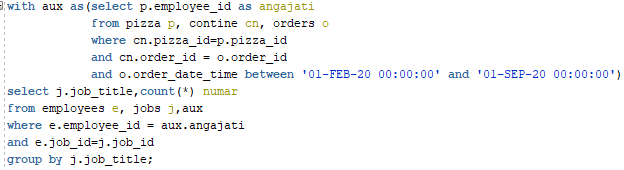
and e.salary> (select avg(j.min\_salary)

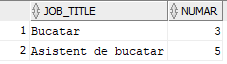
from jobs j

where upper(job\_title)='CHELNER'))

order by c.customer\_id;

1. Afisati numarul bucatarilor si al asistentilor de bucatari care au pregatit pizza pentru comenzi plasate in 2020 intre lunile februarie si august, grupat in functie de numele jobului.





with aux as(select p.employee\_id as angajati

from pizza p, contine cn, orders o

where cn.pizza\_id=p.pizza\_id

and cn.order\_id = o.order\_id

and o.order\_date\_time between '01-FEB-20 00:00:00' and '01-SEP-20 00:00:00')

select j.job\_title,count(\*) numar

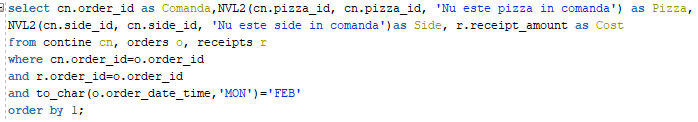
from employees e, jobs j,aux

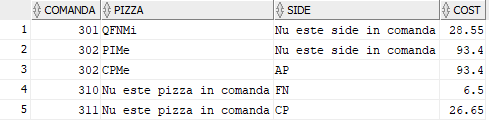
where e.employee\_id = aux.angajati

and e.job\_id=j.job\_id

group by j.job\_title;

1. Afisati codul comenzilor, codul pizzei sau un mesaj daca nu exista, codul alimentului sau un mesaj daca nu exista si costul comenzii pentru comenzile plasate in luna februarie.





select cn.order\_id as Comanda,NVL2(cn.pizza\_id, cn.pizza\_id, 'Nu este pizza in comanda') as Pizza,

NVL2(cn.side\_id, cn.side\_id, 'Nu este side in comanda')as Side, r.receipt\_amount as Cost

from contine cn, orders o, receipts r

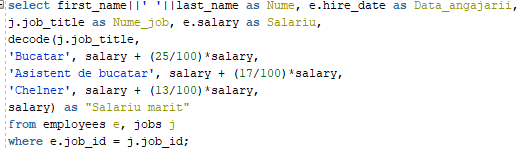
where cn.order\_id=o.order\_id

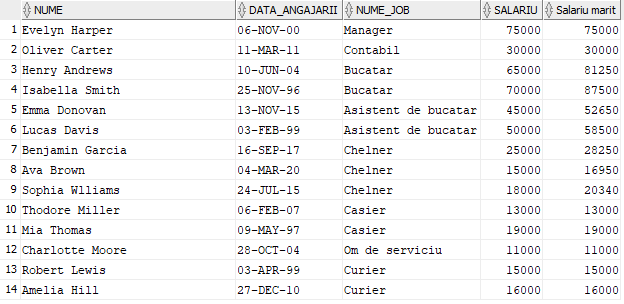
and r.order\_id=o.order\_id

and to\_char(o.order\_date\_time,'MON')='FEB'

order by 1;

1. Să se afişeze numele, data angajării, titlul job-ului, salariul şi o coloană reprezentând salariul după ce se aplică o mărire, astfel: pentru bucatari creşterea este de 25%, pentru asistentii de bucatari creşterea este de 17%, iar salariul chelnerilor creşte cu 13%. Pentru restul salariatilor valoarea nu se modifică.





select first\_name||' '||last\_name as Nume, e.hire\_date as Data\_angajarii,

j.job\_title as Nume\_job, e.salary as Salariu,

decode(j.job\_title,

'Bucatar', salary + (25/100)\*salary,

'Asistent de bucatar', salary + (17/100)\*salary,

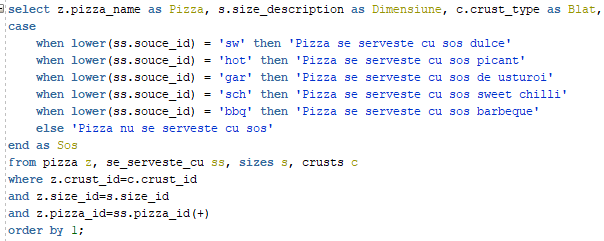
'Chelner', salary + (13/100)\*salary,

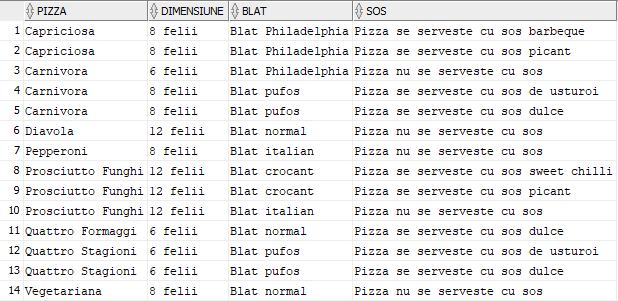
salary) as "Salariu marit"

from employees e, jobs j

where e.job\_id = j.job\_id;

1. Afisati pentru fiecare pizza, numele, dimensiunea, tipul de blat si sosul printr-un mesaj.





select z.pizza\_name as Pizza, s.size\_description as Dimensiune, c.crust\_type as Blat,

case

when lower(ss.souce\_id) = 'sw' then 'Pizza se serveste cu sos dulce'

when lower(ss.souce\_id) = 'hot' then 'Pizza se serveste cu sos picant'

when lower(ss.souce\_id) = 'gar' then 'Pizza se serveste cu sos de usturoi'

when lower(ss.souce\_id) = 'sch' then 'Pizza se serveste cu sos sweet chilli'

when lower(ss.souce\_id) = 'bbq' then 'Pizza se serveste cu sos barbeque'

else 'Pizza nu se serveste cu sos'

end as Sos

from pizza z, se\_serveste\_cu ss, sizes s, crusts c

where z.crust\_id=c.crust\_id

and z.size\_id=s.size\_id

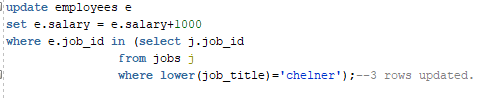
and z.pizza\_id=ss.pizza\_id(+)

order by 1;

EX13. Implementarea a 3 operații de actualizare și de suprimare a datelor utilizând subcereri.

Actualizare:

* Mariti cu 1000 salariul tuturor chelnerilor.



update employees e

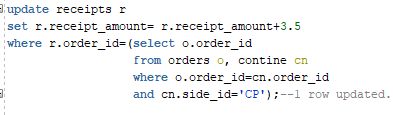
set e.salary = e.salary+1000

where e.job\_id in (select j.job\_id

from jobs j

where lower(job\_title)='chelner');

* Mariti nota de plata cu 3,5 pentru comenzile care contin Choco Pizza.



update receipts r

set r.receipt\_amount= r.receipt\_amount+3.5

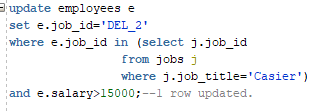
where r.order\_id=(select o.order\_id

from orders o, contine cn

where o.order\_id=cn.order\_id

and cn.side\_id='CP');

* Schimbati codul jobului pentru casierii cu salariu mai mare de 15000.



update employees e

set e.job\_id='DEL\_2'

where e.job\_id in (select j.job\_id

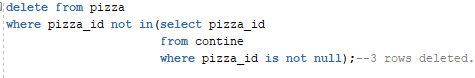
from jobs j

where j.job\_title='Casier')

and e.salary>15000;

Suprimare:

* Stergeti toate pizzele care nu au fost comandate.



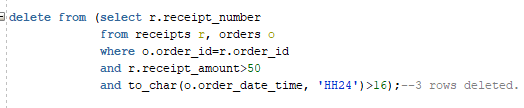
delete from pizza

where pizza\_id not in(select pizza\_id

from contine

where pizza\_id is not null);

* Stergeti toate bonurile care costa mai mult de 50 de lei si au fost emise incepand cu ora 17.



delete from (select r.receipt\_number

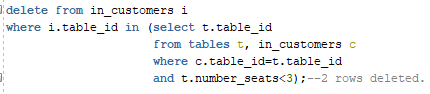
from receipts r, orders o

where o.order\_id=r.order\_id

and r.receipt\_amount>50

and to\_char(o.order\_date\_time, 'HH24')>16);

* Stergeti toti clientii care comanda in restaurant care s-au asezat la mese cu mai putin de 3 locuri.



delete from in\_customers i

where i.table\_id in (select t.table\_id

from tables t, in\_customers c

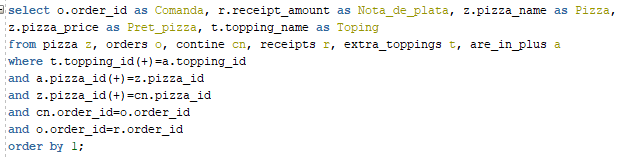
where c.table\_id=t.table\_id

and t.number\_seats<3);

EX16. Formulați în limbaj natural și implementați în SQL: o cerere ce utilizează operația outer-join pe minimum 4 tabele și două cereri ce utilizează operația division.

Outer-join:

Afisati pentru fiecare comanda, id-ul, nota de plata, numele pizzei, pretul pizzei si topingul extra pe care il are.



select o.order\_id as Comanda, r.receipt\_amount as Nota\_de\_plata, z.pizza\_name as Pizza,

z.pizza\_price as Pret\_pizza, t.topping\_name as Toping

from pizza z, orders o, contine cn, receipts r, extra\_toppings t, are\_in\_plus a

where t.topping\_id(+)=a.topping\_id

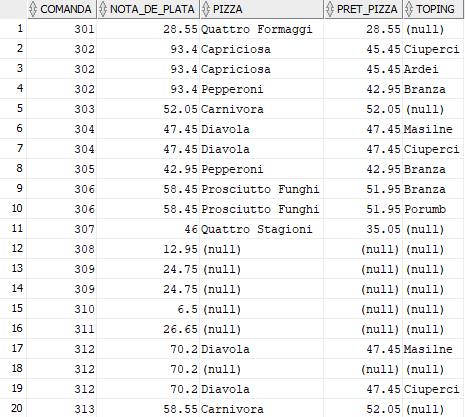
and a.pizza\_id(+)=z.pizza\_id

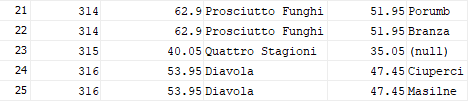
and z.pizza\_id(+)=cn.pizza\_id

and cn.order\_id=o.order\_id

and o.order\_id=r.order\_id

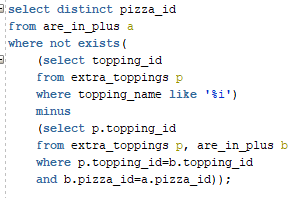
order by 1;





Division:

Sa se obtina codul pizzelor care contin toate topingurile extra care se termina cu i.



select distinct pizza\_id

from are\_in\_plus a

where not exists(

(select topping\_id

from extra\_toppings p

where topping\_name like '%i')

minus

(select p.topping\_id

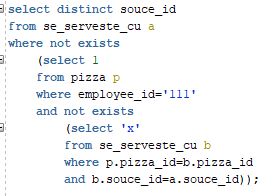
from extra\_toppings p, are\_in\_plus b

where p.topping\_id=b.topping\_id

and b.pizza\_id=a.pizza\_id));



Sa se obtina codul sosurilor care se servesc cu toate pizzele pregatite de angajatul cu codul 111.



select distinct souce\_id

from se\_serveste\_cu a

where not exists

(select 1

from pizza p

where employee\_id='111'

and not exists

(select 'x'

from se\_serveste\_cu b

where p.pizza\_id=b.pizza\_id

and b.souce\_id=a.souce\_id));

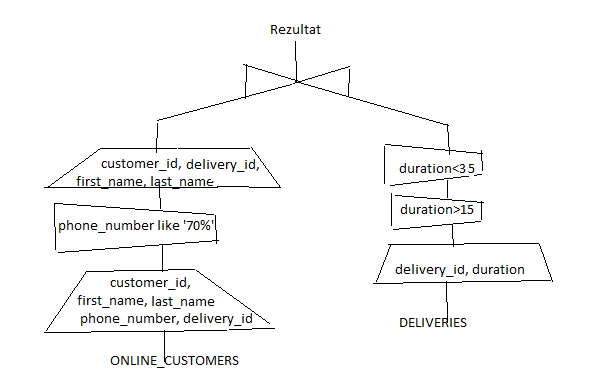


EX17. Optimizarea unei cereri, aplicând regulile de optimizare ce derivă din proprietățile operatorilor algebrei relaționale. Cererea va fi exprimată prin expresie algebrică, arbore algebric și limbaj (SQL), atât anterior cât și ulterior optimizării.

**CERERE:** Sa se afiseze codul, numele de familie, prenumele, codul livrarii si durata pentru clientii a caror numar de telefon incepe cu ‘70’ si ca caror livrare dureaza inte 15 si 35 de minute.

**ANTERIOR:**

Arbore algebric:

****

Expresie algebrica:

R1=PROJECT (ONLINE\_CUSTOMERS, customer\_id, first\_name, last\_name, phone\_number, delivery\_id)

R2=SELECT (R1, phone\_number like ‘70%’)

R3=PROJECT (R2, customer\_id, first\_name, last\_name, delivery\_id)

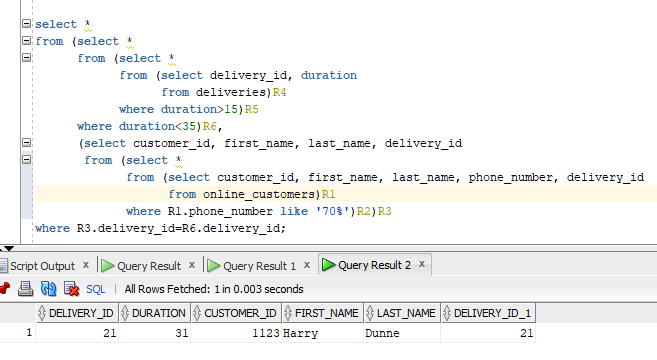
R4=PROJECT (DELIVERIES, delivery\_id, duration)

R5=SELECT (R4, duration>15)

R6=SELECT (R5, duration<35)

Rezultat=JOIN (R6, R3)

SQL:



select \*

from (select \*

from (select \*

from (select delivery\_id, duration

from deliveries)R4

where duration>15)R5

where duration<35)R6,

(select customer\_id, first\_name, last\_name, delivery\_id

from (select \*

from (select customer\_id, first\_name, last\_name, phone\_number, delivery\_id

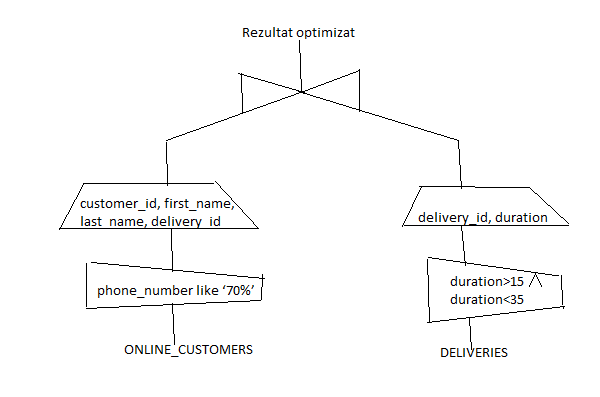
from online\_customers)R1

where R1.phone\_number like '70%')R2)R3

where R3.delivery\_id=R6.delivery\_id;

**ULTERIOR:**

Arbore algebric:

****

Expresie algebrica:

R1=SELECT (ONLINE\_CUSTOMERS, phone\_number like ‘70%’)

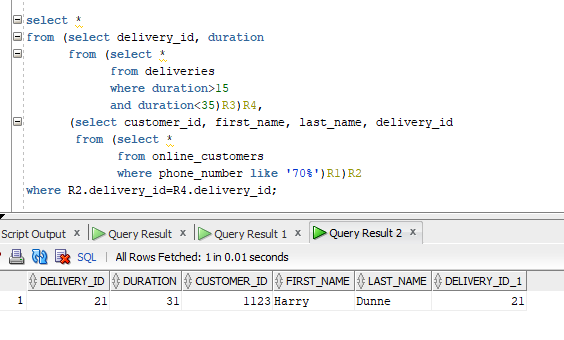
R2=PROJECT (R1, customer\_id, first\_name, last\_name, delivery\_id)

R3=SELECT (DELIVERIES, duration>15 and duration<35)

R4=PROJECT (R3, delivery\_id, duration)

Rezultat=JOIN (R4, R2)

SQL:

****

select \*

from (select delivery\_id, duration

from (select \*

from deliveries

where duration>15

and duration<35)R3)R4,

(select customer\_id, first\_name, last\_name, delivery\_id

from (select \*

from online\_customers

where phone\_number like '70%')R1)R2

where R2.delivery\_id=R4.delivery\_id;

EX18.

1. Realizarea normalizării BCNF, FN4, FN5.

Prin definiție, tabelul este considerat formă normală **Boyce-Codd**, dacă este deja în a treia formă normală și pentru fiecare dependență funcțională între A și B, A ar trebui să fie o super cheie.

Voi ilustra un exemplu folosind relatia dintre client, comanda si nota de plata

Nota de plata

Comanda

Client

{nota de plata, comanda}🡪 client

comanda🡪client

Rezulta:

R1(comanda#, client)

R2(comanda#, nota de plata#)

Forma nomala **FN4** elimină redundanţele datorate relaţiilor m:n, adică datorate dependenţei multiple.

O relație este în a patra formă nomală dacă și numai dacă este în BCNF și nu conține relații m:n independente.

Voi ilustra un exemplu folosind tabelul PIZZA (pizza\_id#, souce\_id, topping\_id). O pizza poate fi servita cu mai multe sosuri si poate avea mai multe topinguri extra, deci souce\_id, topping\_id sunt multidependente de pizza\_id.

Pizza\_id# 🡪 🡪 souce\_id

Pizza\_id# 🡪 🡪 topping\_id

Pentru a aduce relatia in FN4 o voi descompune prin proiectie in 2 relatii:

PIZZA1 (pizza\_id#, souce\_id#)

PIZZA2 (pizza\_id#, toppings\_id#)

PIZZA=join (PIZZA1, PIZZA2)

Forma normala **FN5** îşi propune eliminarea redundanţelor care apar în relaţii m:n dependente.

O relaţie R este în forma normală 5 daca relaţia este în FN4 si nu conţine dependenţe ciclice.

Presupune eliminarea dependentelor join din cadrul relatiilor aflate în FN4.

Fie tabelul ORDER (order\_id, pizza\_id, side\_id)

Presupun ca intre multimile de atribute order\_id, pizza\_id si side\_id exista o dependenta join atunci cand exista dependente multivaloare între fiecare dintre perechile de multimi: (order\_id, pizza\_id), (pizza\_id, side\_id) si (pizza\_id, side\_id).

Se descompune relatia initiala, în scopul obtinerii FN5, trei relatii: r1(order\_id, pizza\_id), r2(pizza\_id, side\_id) si r3(order\_id, side\_id).

|  |  |  |
| --- | --- | --- |
| Order\_id | Pizza\_id | Side\_id |
| 301 | ‘QFNMi’ | ‘CC’ |
| 302 | ‘PFIMe’ | ‘CC’ |
| 302 | ‘QFNMi’ | ‘FN’ |
| 302 | ‘QFNMi’ | ‘CC’ |

|  |  |
| --- | --- |
| Order\_id | Pizza\_id |
| 301 | ‘QFNMi’ |
| 302 | ‘PFIMe’ |
| 302 | ‘QFNMi’ |

|  |  |
| --- | --- |
| Pizza\_id | Side\_id |
| ‘QFNMi’ | ‘CC’ |
| ‘PFIMe’ | ‘CC’ |
| ‘QFNMi’ | ‘FN’ |

|  |  |
| --- | --- |
| Order\_id | Side\_id |
| 301 | ‘CC’ |
| 302 | ‘CC’ |
| 302 | ‘FN’ |

1. Aplicarea denormalizării, justificând necesitatea acesteia.

Denormalizarea unei baze de date reprezinta procesul invers operatiei de normalizare si duce la cresterea redundantei datelor. Prin aceasta se doreste, in principal, cresterea performantei si simplificarea programelor de interogare a datelor.

Denormalizarea se face numai dupa o normalizare corecta

Denormalizarea se face printr-o selectare strategica a structurilor care aduc avantaje semnificative.

Denormalizarea trebuie insotita de masuri suplimentare de asigurare a integritatii datelor.

Obiectivul denormalizării constă în reducerea numărului de join-uricare trebuie efectuate pentru rezolvarea unei interogări, prin realizarea unora dintre acestea în avans, ca făcând parte din proiectarea bazei de date.

In exemplul prezentat pentru FN4 descompunem tabelul PIZZA (pizza\_id#, souce\_id, topping\_id) prin proiectie in 2 relatii PIZZA! SI PIZZA2. Sa presupunem ca majoritatea rapoartelor cerute de conducerea pizzeriei se refera la pizzele produse. In acest caz este cel mai util sa avem toate informatiile despre pizza intr-un singur tabel fiind mult mai costisitor sa facem join intre PIZZA1 si PIZZA2 pentru a afla aceste date. Prin urmare, în acest caz putem prefera varianta denormalizată, cea aflata in FN3.