**UNIVERSITATEA DIN BUCURESTI**

**FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ**

**PROIECT BAZE DE DATE**

**BAZA DE DATE PENTRU GESTIUNEA INFORMAȚIILOR DESPRE RESTAURNANT**

**STUDENT:**

**LASCU GEORGE ALBERT grupa 264**

**BUCUREȘTI 2022**

**CUPRINS**

Introducere...............................................................................................................3

Reguli........................................................................................................................3

Diagrama entitate-relație.........................................................................................4

Diagrama conceptuală..............................................................................................6

Tabelul DEPARTMENTS.............................................................................................7

Tabelul EMPLOYEES..................................................................................................8

Tabelul CLIENTS........................................................................................................9

Tabelul OREDERS....................................................................................................10

Tabelul PRODUCTS..................................................................................................11

Tabelul DETAILS\_ORRDER.......................................................................................13

Tabelul PROVIDER...................................................................................................14

Tabelul RAW\_MATERIALS.......................................................................................14

Tabelul PRODUCT\_SHEET.......................................................................................16

Tabelul TRANSPORT................................................................................................16

**INTRODUCERE:**

De-a lungului timpul technologia a evoluat cum nimeni nu se aștepta. Un bun exemplu este baza de date a unei instituții, companii, etc. Informațiile erau stocate în dosare care mai apoi erau puse în depozite. Acum nu se mai pune problema ca se pot pierde informații cu trecerea timpului. Cu câteva apăsări de buton informațiile pot fi accesate fără probleme.

Modelul reprezintă evidența comenzilor, a angajaților și a clienților existenți în restaurant. Clienții organizați în tabela “CLIENTS” trimit, prin intermediul unui angajat care face dintr-un anumit departament, comenzi de produse organizate in tabela „PRODUCTS”. Aceste produse sunt compuse din materii prime care la randul lor sunt achiziționate de la furnizori organizați în tabela „PROVIDERS”.

Cu ajutorul acestui model conducerea restaurantului poate vizualiza în orice moment câmpurile tabelelor. O dată ce o nouă comandă este plasată, sau un nou angajat se alătură restaurantului, sau retaurantul primește o noua materie primă tabelele sunt modificate.

Cardinalitățile sunt explicate cu ajutorul regulilor.

**REGULI:**

* Într-un departament pot lucra oricâți angajați.
* Un angajat poate lucra într-un singur departament.
* Un client poate avea oricâte comenzi.(are 0 în cazul în care este un client nou)

comanda poate fi data de un singur client.

* Comanda este preluată de un singur angajat.
* Angajatul poate sa nu primească comenzi.(de exemplu un angajat de la departamentul de contabilitate)
* Produsele sunt formate din cel puțin o materie primă.
* Se poate ca o materie primă sa nu facă parte dintr-un produs.
* Furnizorul trebuie să aducă minim o materie primă, iar materia primă nu trebuie neapărat adusă de un furnizor.

**Diagrama entitate-relație**

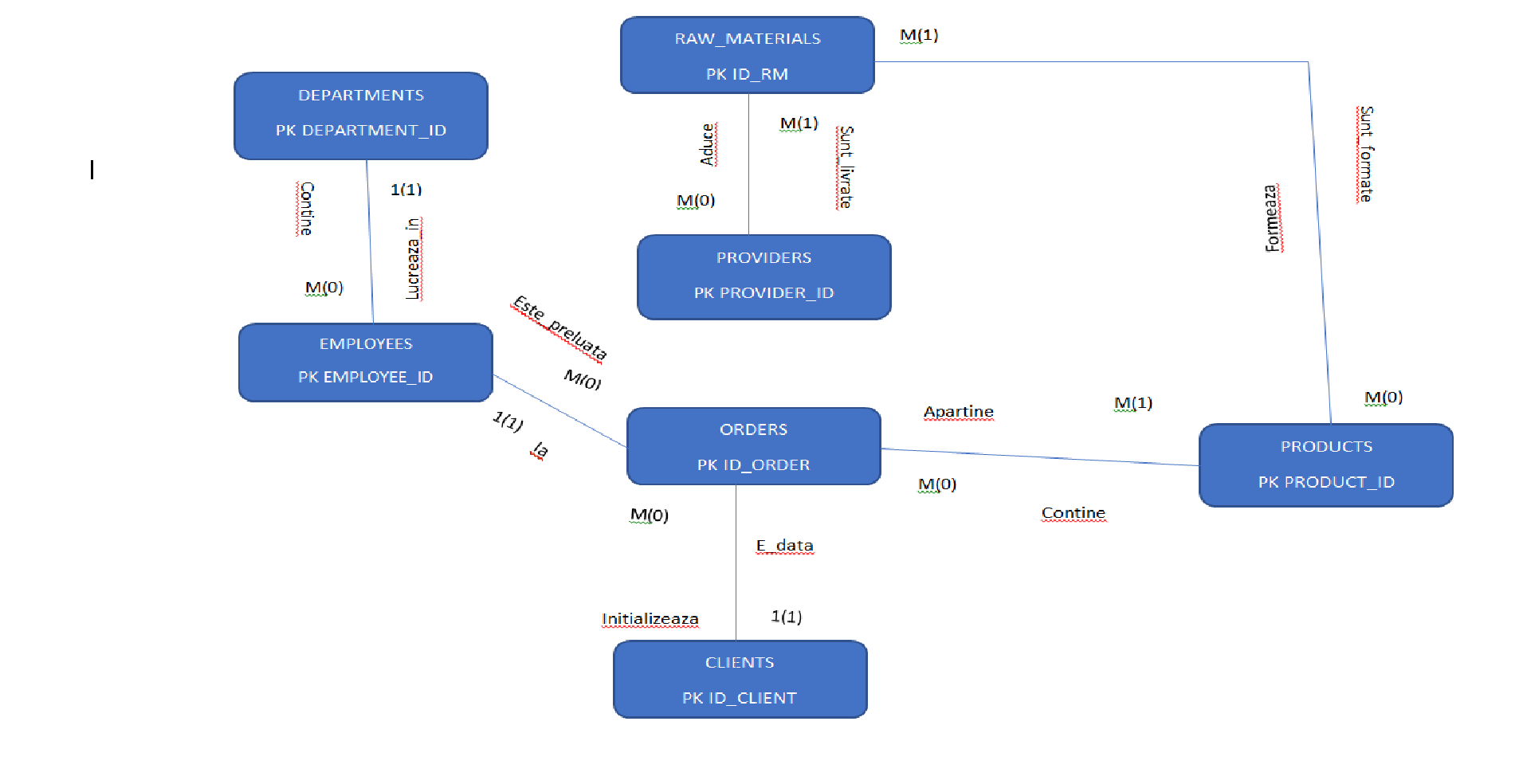
****

Diagrama entitate-relație reprezintă implementarea modelului prezentat mai sus. Diagrama este formată din tabele(entități), cardinalități și verbe. Tabelele sunt: DEPARTMENTS, EMPLOYEES, ORDERS, CLIENTS, PRODUCTS, RAW\_MATERIALS, PROVIDERS.

Tabela DEPARTMENTS reține numele departamentelor existente în care lucrează angajații, dar și un cod unic prin care pot fi diferențiate.

Entitatea EMPLOYEES reprezintă un angajat care este definit prin nume, prenume, data de naștere, data angajării, codul departamentului în care lucrează(cheie externă) și codul său unic.

Entitatea CLIENTS prezintă toți clienții care au venit la restaurant. Ei sunt evidențiați prin codul unic. În plus, tabela poate include și numele sau numărul de telefon, dar nu e obligatoriu.

Entitatea ORDERS reține toate comenzile care au fost date. Această tabela conține si clientul care a dat comanda, cât și angajatul care s-a ocupat de preluarea acesteia. În plus, mai conține si data la care a fost dată comanda.

Entitatea PRODUCTS reprezintă produsele din care este alcătuită o comandă(ex:pizza, burger, prajitură). Mai are doua coloane care ajută restaurantul sa țină evidența la câte produse s-au vândut din câte.

Entitatea RAW\_MATERIAL reprezintă materiile prime cu ajutorul cărora sunt pregătite produsele.

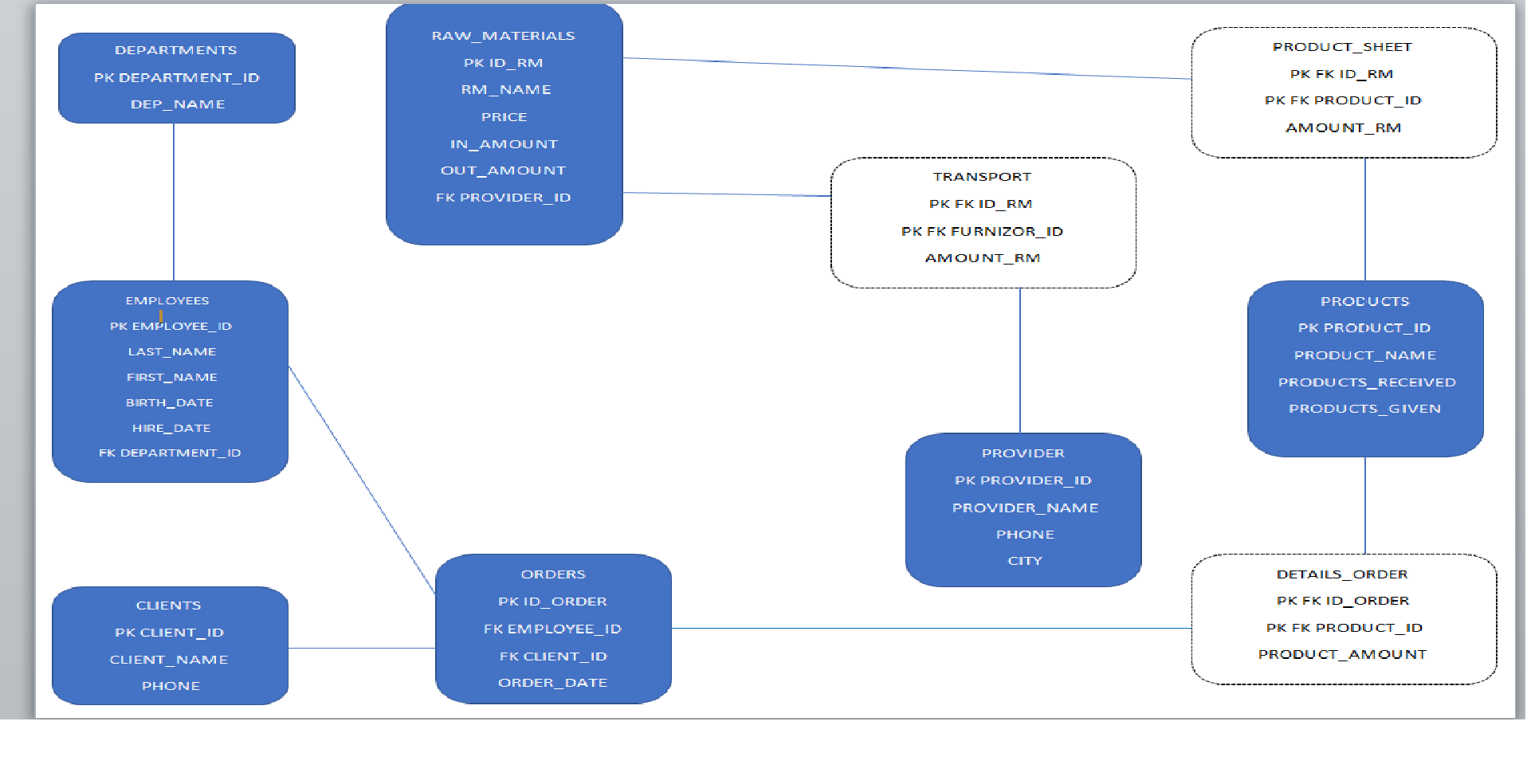
Entitate PROVIDERS prezintă numele si codul furnizorilor care aduc materiile prime.

După cum s-a observat, fiecare entitate are o coloană pentru un id unic. Această coloana se numeste cheie primară și are rolul de a ajuta la căutarea unei informații într-un timp cât mai eficient. În timp ce cheia externă este acea coloană care în alt tabel este cheie primară.

Relațiile sunt verbe care ajută la legarea tabelelor și arată prin ce mod cele doua tabele sunt legate. De exemplu:

* Un angajat lucreaza într-un departament, în timp ce departamentele conțin angajați.
* Comanda este preluată de un angajat, este dată de un client și conține produse.
* Angajatul merge să ia o comandă.
* Clientul inițializează o comandă.
* Produsele aparțin de niște comenzi și sunt formate din materii prime.
* Furnizorul aduce materii prima.

**Digrama conceptuală**



Implementarea în SGBD se va face pe baza acestei diagrame. Diferențele dintre diagrama conceptuală și cea entitate-relație sunt: nu mai sunt prezente cardinalitățile, verbele dispar, apar toate atributele unui tabel și mai apar si tabelele asociative, PRODUCT\_SHEET, TRANSPORT, DETAILS\_ORDER, între entitățile care aveau cardinalități M-M.

Pentru a ține evidența cheilor primare am folosit comanda auto\_increment ca să crească cu unu de fiecare data cand introduc o valoare. Comanda este folositoare, deoarece cel care gestionează baza de date nu are cum să îăi aducă aminte de fiecare data ce numere a pus, astfel doar se introduce null la valoare și se incrementează singură.

Toate datele care au constrângeri de tip NOT NULL sunt obligate să nu fie ramase fară nume, valoare, dată.

**Tabelul DEPARTMENTS**

Creare:

create table `DEPARTMENTS`(

`DEPARTMENT\_ID` INT primary key auto\_increment,

`DEP\_NAME` VARCHAR(25) NOT NULL

)

Introducerea datelor:

insert into departments

values(NULL,'CONTABILITATE')

insert into departments

values(NULL,'VANZARI')

insert into departments

values(NULL,'BUCATARIE')

insert into departments

values(NULL,'ADMINISTRATIV')

insert into departments

values(NULL,'MARKETING')

Coloana DEP\_NAME reprezintă numele departamentului și este de tipul NOT NULL, deoarece toate departamentele trebuie sa aibe nume. Diferențierea între departamente se face prin cheia primară DEPARTMENT\_ID.

**Tabelul EMPLOYEES**

Creare:  
CREATE TABLE `employees`(

`EMPLOYEE\_ID` INT primary key auto\_increment,

`LAST\_NAME` VARCHAR(45) NOT NULL,

`FIRST\_NAME` VARCHAR(45) NOT NULL,

`BIRTH\_DATE` DATE NOT NULL,

`HIRE\_DATE` DATE NOT NULL,

`DEPARTMENT\_ID` INT NOT NULL,

FOREIGN KEY(`DEPARTMENT\_ID`) REFERENCES `DEPARTMENTS` (`DEPARTMENT\_ID`) ON DELETE CASCADE

);

Introducerea datelor:

insert into EMPLOYEES

values(NULL,'Popescu','Alex','97-3-23','20-5-3',1)

insert into EMPLOYEES

values(NULL,'Ilie','Dan','95-6-11','21-12-5',2)

insert into EMPLOYEES

values(NULL,'Dinca','Ionut','95-7-12','21-11-6',2)

insert into EMPLOYEES

values(NULL,'Georgescu','Maria','93-11-3','08-12-5',3)

insert into EMPLOYEES

values(NULL,'Marica','Stefan','98-4-13','10-4-11',3)

insert into EMPLOYEES

values(NULL,'Ratoi','Radu','98-7-23','11-7-31',4)

insert into EMPLOYEES

values(NULL,'Magheru','Cosmin','00-11-15','15-9-1',5)

insert into EMPLOYEES

values(NULL,'Marcel','Petrica','97-6-1','20-6-1',2)

Orice angajat are un nume, prenume, o data a angajării si o data de naștere ca nu au voie sa lipsească. De asemenea, este prezentă și cheia externa DEPARTMENT\_ID care ne arată din ce department face parte angajatul. Cheia externă este de tipul *on delete cascade,* deoarece nu pot exista angajați fară departament.

**Tabelul CLIENTS**

Creare:

create table `CLIENTS`(

`CLIENT\_ID` INT primary key auto\_increment,

`CLIENT\_NAME` VARCHAR(25),

`PHONE` INT UNIQUE

)

Introducerea datelor:

insert into clients

values(null,'Patrascu',0759274590)

insert into clients

values(null,'Gabi',null)

insert into clients

values(null,null,0792614820)

insert into clients

values(null,null,null)

insert into clients

values(null,null,null)

Cum orice restaurant are nevoie de client ca să meargă, aceștia sunt reținuți dupa codul unic, dar e la alegerea lor daca vor să se prezinte dupa nume sau număr de telefon(în cauză că vor să comande online). În plus, numărul de telefon este unic, deoarece nu există ca doi oameni să aibă același număr de telefon.

**Tabelul ORDERS**

Creare:

create table `ORDERS`(

`ID\_ORDER` INT PRIMARY KEY AUTO\_INCREMENT,

`EMPLOYEE\_ID` INT NOT NULL,

`CLIENT\_ID` INT NOT NULL,

`ORDER\_DATE` DATE NOT NULL,

FOREIGN KEY (`EMPLOYEE\_ID`) REFERENCES `EMPLOYEES` (`EMPLOYEE\_ID`),

FOREIGN KEY (`CLIENT\_ID`) REFERENCES `CLIENTS` (`CLIENT\_ID`)

)

Introducerea datelor:

insert into orders

values(null,7,5,'21-6-21')

insert into orders

values(null,8,1,'22-1-7')

insert into orders

values(null,2,3,'21-12-24')

Tabelul ține evidența fiecărei comenzi. Este important să știm ce client a dat-o și ce angajat a preluat-o. De asemenea, se reține si ziua in care a fost plasată.

**Tabelul PRODUCTS**

Creare:

CREATE TABLE PRODUCTS(

PRODUCT\_ID INT PRIMARY KEY AUTO\_INCREMENT,

PRODUCT\_NAME VARCHAR(25) NOT NULL,

AMOUNT\_RECEIVED INT NOT NULL,

AMOUNT\_GIVEN INT NOT NULL

)

alter table products

add constraint check(amount\_given<=amount\_received);

Introducerea datelor:

insert into products

values(null,'PIZZA',23,15)

insert into products

values(null,'BURGERS',30,25)

insert into products

values(null,'CAKE',15,7)

insert into products

values(null,'PASTA',50,49)

insert into products

values(null,'LEMONADE',45,40)

Se observă că în acest tabel se memoreaza cantitatea primită de ce la bucătărie și cantitatea vânduta. De asemenea, se mai memoreaza și ce produs e. Apoi am adăugat constrângerea ca să nu pot vinde o cantitatea mai mare decât cea pe care o primesc, deoarece nu ar avea logică fiindcă nu dispun ce acea cantitate.

**Tabelul DETAILS\_ORDER**

Creare

create table `details\_order`(

`ID\_ORDER` INT NOT NULL,

`PRODUCT\_ID` INT NOT NULL,

`PRODUCT\_AMOUNT` INT NOT NULL,

PRIMARY KEY(ID\_ORDER,PRODUCT\_ID),

FOREIGN KEY(`ID\_ORDER`) REFERENCES ORDERS(ID\_ORDER),

FOREIGN KEY(PRODUCT\_ID) REFERENCES PRODUCTS(PRODUCT\_ID)

)

Introducerea datelor

INSERT INTO details\_order

VALUES(3,5,3)

INSERT INTO details\_order

VALUES(1,4,15)

Acest tabel are cheie primară compusă pentru a ne da seama câte și ce produse s-au consumat la o comandă.

**Tabelul PROVIDER**

Creare:

create table `PROVIDER`(

`PROVIDER\_ID` INT PRIMARY KEY AUTO\_INCREMENT,

`PROVIDER\_NAME` VARCHAR(25) NOT NULL,

`PHONE` INT NOT NULL,

`CITY` VARCHAR(20) NOT NULL

)

Introducerea datelor:

INSERT INTO PROVIDER

VALUES(null,'RITA',0726491045,'BUCURESTI')

INSERT INTO PROVIDER

VALUES(null,'ALPHA',0726445678,'CONSTANTA')

INSERT INTO PROVIDER

VALUES(null,'CARTO',0798765678,'BUZAU')

Am făcut acest tabel pentru a putea memora datele furnizorului nostru.

**Tabelul RAW\_MATERIALS**

Creare:

CREATE TABLE `RAW\_MATERIALS`(

`ID\_RM` INT PRIMARY KEY AUTO\_INCREMENT,

`RM\_NAME` VARCHAR(15) NOT NULL,

`PRICE` INT NOT NULL,

`IN\_AMOUNT` INT NOT NULL,

`OUT\_AMOUNT` INT NOT NULL,

`PROVIDER\_ID` INT,

FOREIGN KEY(PROVIDER\_ID) REFERENCES PROVIDER(PROVIDER\_ID) ON DELETE SET NULL

)

alter table raw\_materials

add constraint check(price>0);

alter table raw\_materials

add constraint check(in\_amount >= out\_amount)

Introducere date:

INSERT INTO raw\_materials

VALUES(null,'FAINA',12,50,47,1)

INSERT INTO raw\_materials

VALUES(null,'oua',2,120,110,2)

INSERT INTO raw\_materials

VALUES(null,'lapte',7,20,20,3)

Acest tabel conține materiile prime primite de la furnizor. De asemenea, este memorată și cantitatea primită, cât și cea livrată. Cheia externă este de tipul *set null*, deoarece dacă renunțăm la un furnizor o să găsim altul în scurt timp, iar acel furnizor va fi pus în locul câmpului care a rămas null. Am adăugat o constrângere care nu mă lasă să folosesc mai multe materii prime decât am si una care obliga pretul sa fie mai mare ca 0.

**Tabelul PRODUCT\_SHEET**

Creare

CREATE TABLE PRODUCT\_SHEET(

ID\_RM INT NOT NULL,

PRODUCT\_ID INT NOT NULL,

AMOUNT\_RM INT NOT NULL,

PRIMARY KEY(ID\_RM,PRODUCT\_ID),

FOREIGN KEY(ID\_RM) REFERENCES RAW\_MATERIALS(ID\_RM),

FOREIGN KEY(PRODUCT\_ID) REFERENCES PRODUCTS(PRODUCT\_ID)

)

Introducerea datelor

INSERT INTO product\_sheet

VALUES(1,3,20)

INSERT INTO product\_sheet

VALUES(2,1,50)

INSERT INTO product\_sheet

VALUES(2,4,35)

Acest tabel ne ajută să vedem din ce și din câte materii prime sunt formate produsele.

**Tabelul TRANSPORT**

Creare

CREATE TABLE `TRANSPORT`(

`ID\_RM` INT NOT NULL,

`PROVIDER\_ID` INT NOT NULL,

`AMOUNT\_RM` INT NOT NULL,

PRIMARY KEY(ID\_RM,PROVIDER\_ID),

FOREIGN KEY(ID\_RM) REFERENCES raw\_materials(ID\_RM),

FOREIGN KEY (PROVIDER\_ID) REFERENCES PROVIDER(PROVIDER\_ID)

)

Introducerea datelor

INSERT INTO TRANSPORT

VALUES(1,1,40)

INSERT INTO TRANSPORT

VALUES(2,2,120)

Tabel construit pentru a vedea ce materie primă este livrată de ce furnizor, dar și cantitatea.