*Proiect Baze de Date*

**Baza de date pentru gestiunea unui lant bancar**

**Realizat de: MUSCALU DIANA**

**Grupa : 144**

**Anul : I**

**1. Descrierea modelului real, a utilității acestuiași a regulilor de funcționare.**

Voi prezenta modelul de date, restricţiile pe care trebuie să le respecte şi voi construi diagrama E/R corespunzătoare. Voi considera, în abordarea iniţială, anumite situaţii care nu sunt optime, în sensul că pot genera redundanţă, anomalii la reactualizări sau nu permit rezolvarea anumitor interogări asupra modelului. Voi încerca să arăt care sunt deficienţele modelului, situaţiile care le-au generat şi cum pot fi corectate.  
 Modelul de date gestioneaza informatii legate de organizarea si buna fucntionare a bancilor. Fiecare banca are angajati si este impartita pe filiale care au clienti. Fiecare client poate avea unul sau mai multe conturi din care pot realiza diferite tranzactii. In plus un client poate sa realizeze un imprumut care este acordat de catre banca. Filiale au de asemenea un program de functionare. Angajatii sunt si acestia imparititi ei putand fi: directori de banca, manageri sau consultanti.   
 Bancile au un an al infintarii si fiecare banca poate avea mai multe filiale. Acestea la randul lor au adresa, un program de functionare si clienti. Clienti pot apartine mai multor filiale si pot avea mai multe conturi la o banca. Bancile pot acorda imprumuturile, iar acestea trebuie sa apartina unui client. Totodata din cont se pot realiza mai multe tranzactii. Angajatii au fiecare cate un salariu si totodata au si date de contact.  
 O astfel de baza de date este necesara pentru a accesa cu usurinta informatii despre banci, clienti cat si despre conturrile, tranzactiile si imprumuturile gestionate de o banca.

**2.Prezentarea constrângerilor(restricții, reguli) impuse asupra modelului.**

Modelul de date respecta anumite restrictii de functionare:   
- o banca are un nume, o data a infiintari, cel putin o filiala si mai multi angajati   
- fiecare filiala are o adresa, clienti si un program de functionare  
- un client poate apartine de mai multe filiale, poate avea mai multe conturi si poate realiza mai multe imprumuturi bancare  
- dintr-un cont se pot realiza mai multe tranzactii care pot fi atat depuneri cat si retrageri  
- un angajat poate fi director de banca, manager sau consultant  
- directorul de banca coordoneaza managerii si consultantii, managerii coordoneaza consultantii, iar consultantii se ocupa de un numar clienti  
- o banca poatea avea o singura data de infiintare, dar pot fi mai multe banci cu aceasta data  
- o filiala poate avea o singura adresa, dar pot fi mai multe filiale cu aceeasi adresa in cazul in care doua banci au sediul in aceeasi cladire  
- un angajat poate avea un singur salariu  
- formatul datelor este de tipul ‘dd.mm.yyyy’

**3.Descrierea entităților, incluzând precizarea cheii primare.**

Pentru modelul de date referitor la gestiunea unei banci avem structurile: BANCA, FILIALA, PROGRAM, CLIENT, CONT, TRANZACTIE, IMPRUMUT, ANGAJAT, SALARIU, CONTACT care reprezinta entitatile.  
 Vom prezenta toate entitatile modelului de date, dand o descriere complete a fiecareia. De asemenea, pentru fiecare entitate se va preciza cheia primara.  
 Toate entitatile care vor fi prezentate sunt independente, cu exceptia subentitatilor corespunzatoare entitatii ANGAJAT si a celor dependente.

BANCA = institutie financiara cu mai multe filiale, ce acorda imprumuturi clientilor, tine evidenta angajatilor si a imprumuturilor pe care le acorda. Cheia primara a entitatii este id\_banca.

FILIALA = institutie controlata de o unitate centrala(banca) ce functioneaza dupa un anumit program si tine evidenta clientilor. Cheia primara a entitatii este id\_filiala.

PROGRAM = entitate dependenta de FILIALA ce retine programul dupa care functioneaza filiala. Cheia primara a entitatii e id\_program.

IMPRUMUT = entitate ce retine detaliile despre un imprumut acordat de banca catre un client. Cheia primara e id\_imprumut.

CLIENT = entitate ce retine datele despre un client al unei filiale, implicit si al unei banci. Cheia primara a entitatii este compusa id\_client

CONT = fiecare client are si un cont care are drept cheie primara id\_cont

TRANZACTIE = entitate ce retine date despre cati bani sunt tranzactionati, data in care s-a intamplat si tipul tranzactiei daca e retragere sau depunere. Cheia primara e id\_tranzactie

ANGAJAT = entitate ce retine date despre fiecare angajat in parte. Acestia au diferite echipe de angajati in subordine: director(manageri+consilieri), manager(consilieri) si consilier(clienti). Cheia primara este: id\_angajat.

DIRECTOR = subentitate a entitatii ANGAJAT ce are in subordine(coordoneaza) un numar de manageri. Cheia primara este: id\_angajat.

MANAGER = subentitate a entitatii ANGAJAT ce are in subordine(coordoneaza) un numar de consilieri. Cheia primara este: id\_angajat.

CONSILIER = subentitate a entitatii ANGAJAT ce se ocupa de un numar de clienti. Cheia primara este: id\_angajat.

SALARIU = entitate dependenta de ANGAJAT ce retine date despre salariul acestuia. Cheia primara este: id\_salariu.

CONTACT = entitate dependenta de ANGAJAT ce retine date despre cum poate fi acesta contactat. Cheia primara este: id\_contact.

**4.Descrierea relațiilor, incluzând precizareacardinalității acestora.**

Vom prezenta relatiile modelului de date, dand o descriere complete pentru fiecare. De fapt, denumirile acestora si legaturile sunt suggestive, reflectand continutul acestora si entitatile pe care le leaga.

BANCA\_are\_FILIALA = relatie care leaga entitatile BANCA si FILIALA, reflecatnd legatura dintre acestea (ce filiale apartin unei banci). Ea are cardinalitatea minima 1:1(o filiala apartine unei banci) si cardinalitatea maxima 1:M(o banca poate avea mai multe filiale, iar o filiala apartine unei singure banci)

BANCA\_angajeaza\_ANGAJAT = relatie care leaga entitatile BANCA si ANGAJAT, reflecatnd legatura dintre acestea(ce angajati lucreaza la banca). Ea are cardinalitatea minina 1:1(un angajat lucreaza la o banca) si maxima 1:M(la o banca pot fi mai multi angajati).

BANCA\_acorda\_IMPRUMUT = relatie care leaga BANCA si IMPRUMUT, reflectand legatura dintre acestea(ce imprumuturi au fost acordate de banca). Ea are cardinalitate minima 1:1(o banca a acordat un imprumut)

FILIALA\_corespunde\_PROGRAM = relatie care leaga entitatile FILIALA si PROGRAM, reflectand legatura dintre acestea(unei filiale ii corespunde un program de functionare). Ea are cardinalitatea minima si maxima 1:1(unei filiale ii corespunde un program).

FILIALA\_apartinere\_CONT = relatie many-to-many (rezolvata prin entitatea asociativa APARTINERE), care leaga entitatile FILIALA si CONT, reflectand legatura dintre acestea(un cont apartine undei filiale). Are cardinalitate minima 1:1(un cont apartine de o filiala) si maxima M:N(un cont poate apratine de mai multe filiale si o filiala poate avea mai multe conturi)

ANGAJAT\_beneficiaza\_SALARIU = relatie care leaga entitatile ANGAJAT si SALARIU, reflectand legatura dintre acestea(un angajat are un salariu). Ea are cardinalitatea minima si maxima 1:1(un angajat are un singur salariu)

ANGAJAT\_detine\_CONTACT= relatie care leaga entitatile ANGAJAT si CONTACT, reflectand legatura dintre acestea(un angajat detine date de contact). Ea are cardinalitatea minima si maxima 1:1(un angajat are un singur email si telefon)

CLIENT\_detine\_CONT = relatie many to many(rezolvata prin crearea entitati associative CLIENT\_CONT) care leaga entitatile CLIENT si CONT, reflectand legatura dintre acestea(un client are cont). Ea are cardinalitate minima 1:1(un client are un cont) si maxima M:N(un client poate avea mai multe conturi, iar un cont poate apartine mai multor clienti)

CONT\_executa\_TRANZACTIE = relatie care leaga entitatile CONT si TRANZACTIE, reflectand legatura dintre acestea(un cont executa tranzactii). Ea are cardinalitate minima 1:1(un cont executa o tranzactie) si maxima 1:M(un cont poate executa mai multe tranzactii)

CLIENT\_face\_IMPRUMUT = relatie care leaga entitatile CLIENT si IMPRUMUT, reflectand legatura dintre acestea(clientul face imprumut). Ea are cardinalitate minima 1:1(un client face un imprumut) si maxima 1:M(un client poate face mai multe imprumuturi, iar un imprumut apartine unui singur client).

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

1. Entitatea BANCA are atributele:   
¤ id\_banca = variabila de tip intreg, de lungime maxima 10 si reprezinta codul bancii  
¤ nume = variabila de tip char, de lungime maxima 60 si reprezinta numele bancii  
¤ an\_infiintare = variabila de tip char, de lungime maxima 60 si reprezinta data infiintarii bancii  
¤ nr\_filiale = variabila de tip intreg, de lungime maxima 10 si reprezinta nr de filiale ale bancii

2. Entitatea FILIALA are atributele:   
¤ id\_filiala = variabila de tip intreg, de lungime maxima 10 si reprezinta codul filialei  
¤ id\_banca = variabila de tip intreg, de lungime maxima 10 si reprezinta codul bancii  
¤ oras = variabila de tip char, de lungime maxima 60 si reprezinta orasul in care se afla filiala  
¤ nr\_clienti = variabila de tip intreg, de lungime maxima 10 si reprezinta nr de lcienti pe care ii are filiala

3. Entitatea PROGRAM are atributele:  
¤ id\_program = variabila de tip intreg, de lungime maxima 10 si reprezinta codul program  
¤ id\_filiala = variabila de tip intreg, de lungime maxima 10 si reprezinta codul filialei   
¤ ora\_deschidere = variabila de tip char, de lungime maxima 60 si reprezinta ora de deschidere a filialei  
¤ ora\_inchidere = variabila de tip char, de lungime maxima 60 si reprezinta ora de inchidere a filialei

4. Entitatea ANGAJAT are atributele:   
¤ id\_angajat = variabila de tip intreg, de lungime maxima 10 si reprezinta codul anagajatului  
¤ id\_banca = variabila de tip intreg, de lungime maxima 10 si reprezinta codul banii   
¤ nume = variabila de tip char, de lungime maxima 60 si reprezinta numele angajatului  
¤ data\_angajari = variabila de tip char, de lungime maxima 60 si reprezinta data de angajare

5. Subentitatea DIRECTOR are atributele:   
¤ nr\_manageri = variabila de tip intreg, de lungime maxima 10 si reprezinta numarul de manageri aflati in subordine

6. Subentitatea MANAGER are atributele:   
¤ nr\_consilieri = variabila de tip intreg, de lungime maxima 10 si reprezinta numarul de consilieri aflati in subordine

7. Subentitatea CONSILIER are atributele:   
¤ nr\_clienti = variabila de tip intreg, de lungime maxima 10 si reprezinta numarul de clienti de care se ocupa

8. Entitatea CLIENT are atributele:  
¤ id\_client = variabila de tip intreg, de lungime maxima 10 si reprezinta codul clientului  
¤ nume = variabila de tip char, de lungime maxima 60 si reprezinta numele clientului  
¤ prenume = variabila de tip char, de lungime maxima 60 si reprezinta prenumele clientului

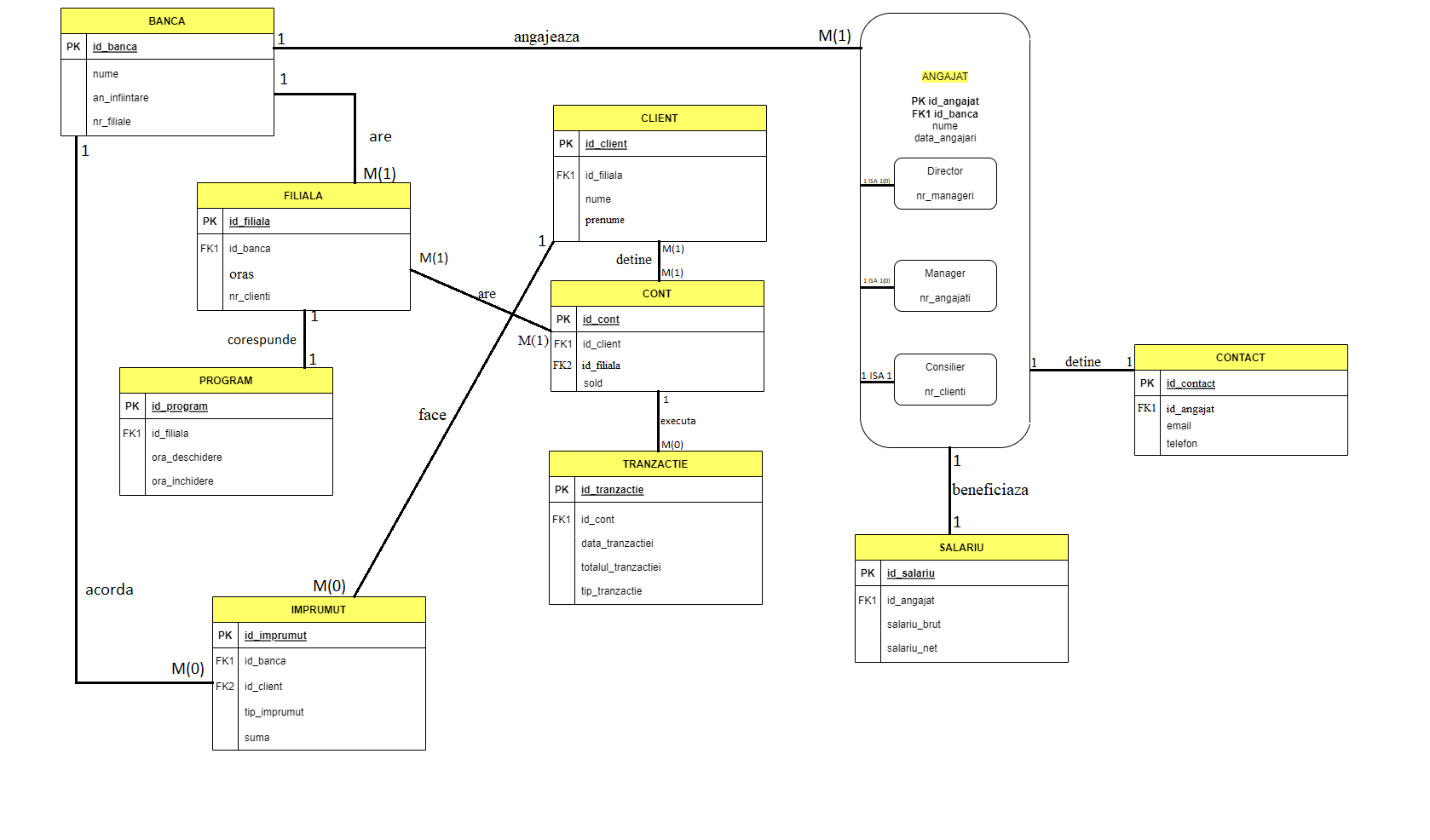
9. Entitatea CONT are atributele:  
¤ id\_cont = variabila de tip intreg, de lungime maxima 10 si reprezinta codul clientului  
¤ sold = variabila de tip intreg, de lungime maxima 10 si reprezinta soldul curent din cont

10. Entitatea TRANZACTIE are atributele:  
¤ id\_tranzactie = variabila de tip intreg, de lungime maxima 10 si reprezinta codul tranzactiei  
¤ id\_cont = variabila de tip intreg, de lungime maxima 10 si reprezinta codul clientului  
¤ data\_tranzactiei = variabila de tip char, de lungime maxima 60 si reprezinta data tranzactiei  
¤ totalul\_tranzactiei = variabila de tip intreg, de lungime maxima 10 si reprezinta totatlul tranzactiei  
¤ tip\_tranzactie = variabila de tip char, de lungime maxima 60 si reprezinta tipul tranzactiei

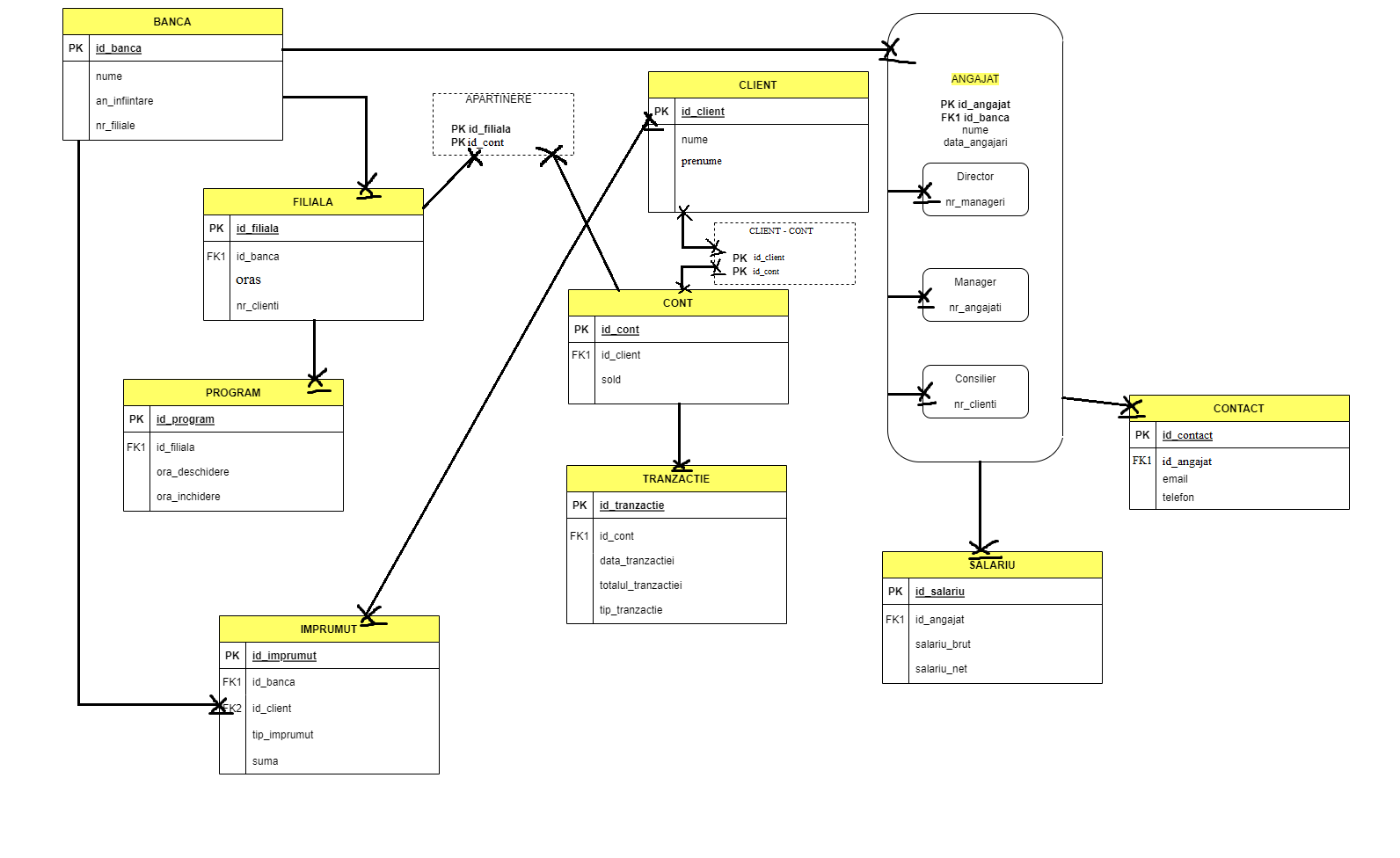
11. Entitatea IMPRUMUT are atributele:  
¤ id\_imprumut = variabila de tip intreg, de lungime maxima 10 si reprezinta codul imprumutului  
¤ id\_banca = variabila de tip intreg, de lungime maxima 10 si reprezinta codul bancii  
¤ id\_client = variabila de tip intreg, de lungime maxima 10 si reprezinta codul clientului  
¤ tip\_imprumut = variabila de tip char, de lungime maxima 60 si reprezinta tipul imprumutului  
¤ suma = variabila de tip intreg, de lungime maxima 10 si reprezinta suma totala a imprumutului

12. Entitatea SALARIU are atributele:  
¤ id\_salariu = variabila de tip intreg, de lungime maxima 10 si reprezinta codul salariului  
¤ id\_angajat = variabila de tip intreg, de lungime maxima 10 si reprezinta codul angajatului  
¤ salariu\_brut = variabila de tip intreg, de lungime maxima 10 si reprezinta salariul brut  
¤ salariu\_net= variabila de tip intreg, de lungime maxima 10 si reprezinta salariul net

13. Entitatea CONTACT are atributele:  
¤ id\_contact = variabila de tip intreg, de lungime maxima 10 si reprezinta codul contactului  
¤ id\_angajat = variabila de tip intreg, de lungime maxima 10 si reprezinta codul angajatului  
¤ email = variabila de tip char, de lungime maxima 60 si reprezinta email-ul  
¤ telefon = variabila de tip char, de lungime maxima 60 si reprezinta tipul telefonului

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

**7. 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.**



**8. Enumerarea schemelor relaționale corespunzătoare diagramei conceptuale proiectate la punctul 7.**

Schemele relaţionale corespunzătoare diagramei conceptuale sunt următoarele:

BANCA(id\_banca#, nume, an\_infiintare, nr\_filiale)  
FILIALA(id\_filiala#, id\_banca#, oras, nr\_clienti)  
PROGRAM(id\_program#, id\_filiala#, ora\_deschidere, ora\_inchidere)  
IMPRUMUT(id\_imprumut#, id\_banca#, id\_client#, tip\_imprumut, suma)  
CLIENT(id\_client#, nume, prenume)  
CONT(id\_cont, sold)  
TRANZACTIE(id\_tranzactie#, id\_cont#, data\_tranzactiei, totalul\_tranzactiei, tip\_tranzactie)  
ANGAJAT(id\_angajat#, id\_banca#, nume, data\_angajarii)  
DIRECTOR(id\_angajat#, nr\_manageri)  
MANAGER(id\_angajat#, nr\_consilieri)  
CONSILIER(id\_ angajat#, nr\_clienti)  
CONTACT(id\_contact#, id\_angajat#, email, telefon)  
SALARIU(id\_salariu#, id\_angajat#, salariu\_brut, salariu\_net)  
APARTINERE(id\_filiala#, id\_cont#)  
CLIENT\_CONT(id\_client#, id\_cont#)

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

FORMA NORMALA 1 (FN1)

* O relatie se afla in FN1, daca fiecarui atriubut care o compune ii corespunde o valoare indivizibila.
* FN1 este si cea care impune faptul ca fiecare inregistrare trebuie sa fie definite astfel incat sa fie identificata unic prin intermediul unei chei primare.

NON FN1 :

|  |  |
| --- | --- |
| id\_client# | nume |
| 1 | YNY Sebi |
| 2 | Toader Cristi |
| 3 | Dascalescu Iulia |

FN1:

|  |  |  |
| --- | --- | --- |
| id\_client# | nume | prenume |
| 1 | YNY | Sebi |
| 2 | Toader | Cristi |
| 3 | Dascalescu | Iulia |

FORMA NORMALA 2 (FN2)

* O relatie se afla in FN2, daca si numai daca aceasta relatie este deja in FN1 si fiecare atribut care nu este cheie primara este dependent de intreaga cheie primara.
* FN2 interzice existenta dependentelor functionale in cadrul relatiei.

NON FN2 :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id\_angajat# | id\_banca# | nume | prenume | data\_angajarii | denumire\_banca |
| 1 | 100 | Popescu | Dan | 12.05.2000 | Transilvania |
| 2 | 101 | Cazacu | Alina | 10.07.2020 | Raiffeisen |
| 3 | 102 | Conte | Carina | 01.06.2022 | BRD |

FN2 :

|  |  |
| --- | --- |
| id\_banca# | denumire\_banca |
| 100 | Transilvania |
| 101 | Raiffeisen |
| 102 | BRD |

FORMA NORMALA 3 (FN3)

* O relatie se afla in a treia forma (FN3), daca si numai daca aceasta relatie este deja in FN2 si fiecare atribut care nu este cheie, depinde direct de cheia primara.

NON FN3 :

|  |  |  |  |
| --- | --- | --- | --- |
| id\_cont# | sold | nume | prenume |
| 00 | 100000 | YNY | Sebi |
| 01 | 2000 | Toader | Cristi |
| 02 | 50000 | Dascalescu | Iulia |

* In tabelul de mai sus putem observa cum numele si prenumele nu au nici o legtaura directa cu cheia primara a tabelului si anume id\_cont#.

FN3 :

|  |  |
| --- | --- |
| id\_cont# | sold |
| 00 | 100000 |
| 01 | 2000 |
| 02 | 50000 |

**10. 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).**

*Partea de cod se afla in notepad-ul: Muscalu\_Diana-CREARE\_INSERARE.*

--------------------------BANCA--------------------------

CREATE TABLE BANCA (

id\_banca NUMBER(10) PRIMARY KEY,

nume VARCHAR2(60) NOT NULL UNIQUE,

an\_infiintare DATE NOT NULL,

filiale NUMBER(10)

CONSTRAINT chk\_banca CHECK (filiale > 0)

);

INSERT INTO BANCA (id\_banca, nume, an\_infiintare, filiale)

VALUES (100, 'Banca Transilvania', to\_date('01/12/1993','DD/MM/YYYY'), 10);

INSERT INTO BANCA (id\_banca, nume, an\_infiintare, filiale)

VALUES (101, 'Raiffeisen Bank', to\_date('01/06/2002','DD/MM/YYYY'), 20);

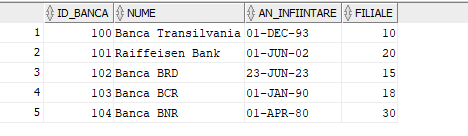
INSERT INTO BANCA (id\_banca, nume, an\_infiintare, filiale)

VALUES (102, 'Banca BRD', to\_date('23/06/1923','DD/MM/YYYY'), 15);

INSERT INTO BANCA (id\_banca, nume, an\_infiintare, filiale)

VALUES (103, 'Banca BCR', to\_date('01/01/1990','DD/MM/YYYY'), 18);

INSERT INTO BANCA (id\_banca, nume, an\_infiintare, filiale)

VALUES (104, 'Banca BNR', to\_date('01/04/1880','DD/MM/YYYY'), 30);  
  


--------------------------FILIALA--------------------------

CREATE TABLE FILIALA (

id\_filiala NUMBER(10) UNIQUE NOT NULL,

id\_banca NUMBER(10) NOT NULL,

CONSTRAINT pk\_fil PRIMARY KEY(id\_filiala, id\_banca),

CONSTRAINT fkey\_fil\_banca FOREIGN KEY (id\_banca) REFERENCES BANCA(id\_banca) ON DELETE SET NULL,

oras VARCHAR2(60) NOT NULL,

nr\_clienti NUMBER(10)

CONSTRAINT chk\_clienti CHECK (nr\_clienti >=0)

);

ALTER TABLE FILIALA

ADD CONSTRAINT fkey\_fil\_banca FOREIGN KEY (id\_banca) references BANCA(id\_banca);

INSERT INTO FILIALA (id\_filiala, id\_banca, oras, nr\_clienti)

VALUES (1, 100, 'BUCURESTI', 10000);

INSERT INTO FILIALA (id\_filiala, id\_banca, oras, nr\_clienti)

VALUES (2, 100, 'TIMISOARA', 15000);

INSERT INTO FILIALA (id\_filiala, id\_banca, oras, nr\_clienti)

VALUES (3, 101, 'BRASOV', 20000);

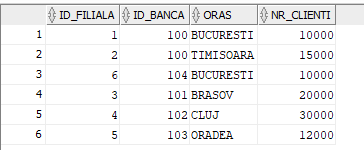
INSERT INTO FILIALA (id\_filiala, id\_banca, oras, nr\_clienti)

VALUES (4, 102, 'CLUJ', 30000);

INSERT INTO FILIALA (id\_filiala, id\_banca, oras, nr\_clienti)

VALUES (5, 103, 'ORADEA', 12000);

INSERT INTO FILIALA (id\_filiala, id\_banca, oras, nr\_clienti)

VALUES (6, 104, 'BUCURESTI', 10000);  
  


--------------------------PROGRAM--------------------------  
CREATE TABLE PROGRAM (

id\_program NUMBER(10) PRIMARY KEY,

id\_filiala NUMBER(10) NOT NULL,

CONSTRAINT fkey\_prog\_fil FOREIGN KEY (id\_filiala) REFERENCES FILIALA(id\_filiala) ON DELETE SET NULL,

ora\_deschidere VARCHAR2(60) NOT NULL,

ora\_inchidere VARCHAR(60) NOT NULL

);

INSERT INTO PROGRAM(id\_program, id\_filiala, ora\_deschidere, ora\_inchidere)

VALUES (1, 1, '10:00', '18:00');

INSERT INTO PROGRAM(id\_program, id\_filiala, ora\_deschidere, ora\_inchidere)

VALUES (2, 2, '08:00', '16:00');

INSERT INTO PROGRAM(id\_program, id\_filiala, ora\_deschidere, ora\_inchidere)

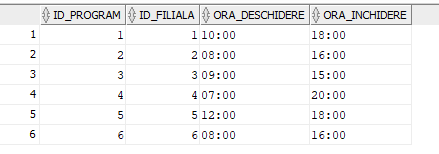
VALUES (3, 3, '09:00', '15:00');

INSERT INTO PROGRAM(id\_program, id\_filiala, ora\_deschidere, ora\_inchidere)

VALUES (4, 4, '07:00', '20:00');

INSERT INTO PROGRAM(id\_program, id\_filiala, ora\_deschidere, ora\_inchidere)

VALUES (5, 5, '12:00', '18:00');

INSERT INTO PROGRAM(id\_program, id\_filiala, ora\_deschidere, ora\_inchidere)  
VALUES (6, 6, '08:00', '16:00');  
  


--------------------------IMPRUMUT--------------------------  
CREATE TABLE IMPRUMUT (

id\_imprumut NUMBER(10) PRIMARY KEY,

id\_banca NUMBER(10) NOT NULL,

id\_client NUMBER(10) NOT NULL,

CONSTRAINT fkey\_impr\_ban FOREIGN KEY (id\_banca) REFERENCES BANCA(id\_banca) ON DELETE SET NULL,

CONSTRAINT fkey\_impr\_client FOREIGN KEY (id\_client) REFERENCES CLIENTT(id\_client) ON DELETE SET NULL,

tip\_imprumut VARCHAR2(60) NOT NULL,

suma NUMBER(10)

CONSTRAINT chk\_suma CHECK (suma> 0)

);

INSERT INTO IMPRUMUT(id\_imprumut, id\_banca, id\_client, tip\_imprumut, suma)

VALUES (1, 100, 1, 'prima casa', 25000);

INSERT INTO IMPRUMUT(id\_imprumut, id\_banca, id\_client, tip\_imprumut, suma)  
VALUES (2, 104, 7, 'bancar', 300000);

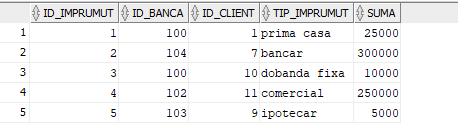
INSERT INTO IMPRUMUT(id\_imprumut, id\_banca, id\_client, tip\_imprumut, suma)

VALUES (3, 100, 10, 'dobanda fixa', 10000);

INSERT INTO IMPRUMUT(id\_imprumut, id\_banca, id\_client, tip\_imprumut, suma)

VALUES (4, 102, 11, 'comercial', 250000);

INSERT INTO IMPRUMUT(id\_imprumut, id\_banca, id\_client, tip\_imprumut, suma)

VALUES (5, 103, 9, 'ipotecar', 5000);  
  


--------------------------CONT--------------------------  
CREATE TABLE CONTT (

id\_cont NUMBER(10) PRIMARY KEY,

sold NUMBER(10)

);

INSERT INTO CONTT (id\_cont, sold)

VALUES (00, 1000);

INSERT INTO CONTT (id\_cont, sold)

VALUES (01, 5000);

INSERT INTO CONTT (id\_cont, sold)

VALUES (02, 10000);

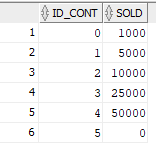
INSERT INTO CONTT (id\_cont, sold)

VALUES (03, 25000);

INSERT INTO CONTT (id\_cont, sold)

VALUES (04, 50000);

INSERT INTO CONTT (id\_cont, sold)

VALUES (05, 0);  
  


--------------------------CLIENT--------------------------

CREATE TABLE CLIENTT (

id\_client NUMBER(10) UNIQUE NOT NULL,

nume VARCHAR2(60) NOT NULL,

prenume VARCHAR(60) NOT NULL

);

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

VALUES (1, 'Voinea', 'Ana');

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

VALUES (2, 'Iosoi', 'Cristiana');

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

VALUES (3, 'Balu', 'Valentin-Ionut');

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

VALUES (4, 'Muscalu', 'Diana');

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

VALUES (5, 'Mocanu', 'Radu');

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

VALUES (6, 'Chiriac', 'Ella');

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

VALUES (7, 'Ionescu', 'Raluca');

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

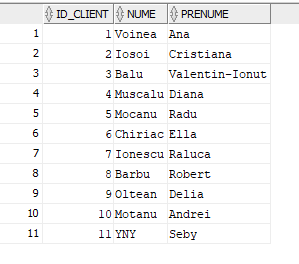
VALUES (8, 'Barbu', 'Robert');

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

VALUES (9, 'Oltean', 'Delia');

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

VALUES (10, 'Motanu', 'Andrei');

INSERT INTO CLIENTT(id\_client, nume, prenume)  
VALUES (11, 'YNY', 'Seby');  
  


--------------------------TRANZACTIE--------------------------  
CREATE TABLE TRANZACTIE (

id\_tranzactie NUMBER(10) PRIMARY KEY,

id\_cont NUMBER(10) NOT NULL,

CONSTRAINT fkey\_tran\_cont FOREIGN KEY (id\_cont) REFERENCES CONTT(id\_cont) ON DELETE SET NULL,

data\_tranzactiei VARCHAR2(60) NOT NULL,

totalul\_tranzactiei NUMBER(10),

tip\_tranzactie VARCHAR2(60) NOT NULL,

CONSTRAINT chk\_total CHECK (totalul\_tranzactiei > 0)

);

INSERT INTO TRANZACTIE (id\_tranzactie, id\_cont, data\_tranzactiei, totalul\_tranzactiei, tip\_tranzactie)

VALUES (1, 00, '01.01.2021', 100, 'depunere');

INSERT INTO TRANZACTIE (id\_tranzactie, id\_cont, data\_tranzactiei, totalul\_tranzactiei, tip\_tranzactie)

VALUES (2, 01, '02.02.2022', 1500, 'retragere');

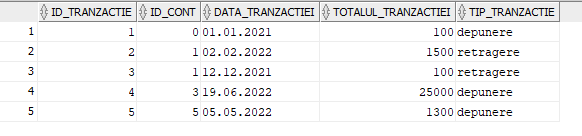
INSERT INTO TRANZACTIE (id\_tranzactie, id\_cont, data\_tranzactiei, totalul\_tranzactiei, tip\_tranzactie)

VALUES (3, 01, '12.12.2021', 100, 'retragere');

INSERT INTO TRANZACTIE (id\_tranzactie, id\_cont, data\_tranzactiei, totalul\_tranzactiei, tip\_tranzactie)

VALUES (4, 03, '19.06.2022', 25000, 'depunere');

INSERT INTO TRANZACTIE (id\_tranzactie, id\_cont, data\_tranzactiei, totalul\_tranzactiei, tip\_tranzactie)

VALUES (5, 05, '05.05.2022', 1300, 'depunere');  
  


--------------------------ANGAJAT--------------------------

CREATE TABLE ANGAJAT (

id\_angajat NUMBER(10) PRIMARY KEY,

id\_banca NUMBER(10) NOT NULL,

CONSTRAINT fkey\_ang\_ban FOREIGN KEY (id\_banca) REFERENCES BANCA(id\_banca) ON DELETE SET NULL,

nume VARCHAR2(60) NOT NULL,

data\_angajari VARCHAR2(60) NOT NULL

);

INSERT INTO ANGAJAT(id\_angajat, id\_banca, nume, data\_angajari)

VALUES (0, 100, 'POPESCU DAN', '12.05.2000');

INSERT INTO ANGAJAT(id\_angajat, id\_banca, nume, data\_angajari)

VALUES (1, 102, 'CAZACU ALINA', '10.07.2020');

INSERT INTO ANGAJAT(id\_angajat, id\_banca, nume, data\_angajari)

VALUES (2, 101, 'MARIN PREDA', '01.01.1995');

INSERT INTO ANGAJAT(id\_angajat, id\_banca, nume, data\_angajari)

VALUES (3, 102, 'CONTE CARINA', '01.06.2022');

INSERT INTO ANGAJAT(id\_angajat, id\_banca, nume, data\_angajari)

VALUES (4, 103, 'JHON MAYER', '31.07.2000');

INSERT INTO ANGAJAT(id\_angajat, id\_banca, nume, data\_angajari)

VALUES (5, 104, 'POPESCU CRISTIAN', '12.12.2012');

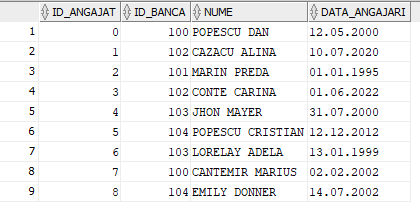
INSERT INTO ANGAJAT(id\_angajat, id\_banca, nume, data\_angajari)

VALUES (6, 103, 'LORELAY ADELA', '13.01.1999');

INSERT INTO ANGAJAT(id\_angajat, id\_banca, nume, data\_angajari)

VALUES (7, 100, 'CANTEMIR MARIUS', '02.02.2002');

INSERT INTO ANGAJAT(id\_angajat, id\_banca, nume, data\_angajari)

VALUES (8, 104, 'EMILY DONNER', '14.07.2002');  
  


--------------------------DIRECTOR--------------------------  
CREATE TABLE DIRECTOR (

id\_angajat NUMBER(10) not null REFERENCES ANGAJAT(id\_angajat),

PRIMARY KEY(id\_angajat),

nr\_manageri NUMBER(10)

);  
INSERT INTO DIRECTOR (id\_angajat)

SELECT id\_angajat

FROM ANGAJAT

WHERE id\_angajat = 0;

UPDATE DIRECTOR SET

nr\_manageri = 5

WHERE id\_angajat = 0;

INSERT INTO DIRECTOR (id\_angajat)

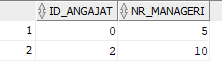
SELECT id\_angajat

FROM ANGAJAT

WHERE id\_angajat = 2;

UPDATE DIRECTOR SET

nr\_manageri = 10

WHERE id\_angajat = 2;  
  


--------------------------MANAGER--------------------------  
CREATE TABLE MANAGER (

id\_angajat NUMBER(10) not null REFERENCES ANGAJAT(id\_angajat),

PRIMARY KEY(id\_angajat),

nr\_consilieri NUMBER(10)

);

INSERT INTO MANAGER (id\_angajat)

SELECT id\_angajat

FROM ANGAJAT

WHERE id\_angajat = 6;

UPDATE MANAGER SET

nr\_consilieri = 4

WHERE id\_angajat = 6;

INSERT INTO MANAGER (id\_angajat)

SELECT id\_angajat

FROM ANGAJAT

WHERE id\_angajat = 7;

UPDATE MANAGER SET

nr\_consilieri = 5

WHERE id\_angajat = 7;

INSERT INTO MANAGER (id\_angajat)

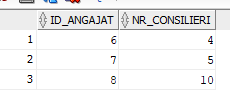
SELECT id\_angajat

FROM ANGAJAT

WHERE id\_angajat = 8;

UPDATE MANAGER SET

nr\_consilieri = 10

WHERE id\_angajat = 8;  
  


--------------------------CONSILIER--------------------------  
CREATE TABLE CONSILIER (

id\_angajat NUMBER(10) not null REFERENCES ANGAJAT(id\_angajat),

PRIMARY KEY(id\_angajat),

nr\_clienti NUMBER(10)

);  
INSERT INTO CONSILIER (id\_angajat)

SELECT id\_angajat

FROM ANGAJAT

WHERE id\_angajat = 1;

UPDATE CONSILIER SET

nr\_clienti = 100

WHERE id\_angajat = 1;

INSERT INTO CONSILIER (id\_angajat)

SELECT id\_angajat

FROM ANGAJAT

WHERE id\_angajat = 3;

UPDATE CONSILIER SET

nr\_clienti = 50

WHERE id\_angajat = 3;

INSERT INTO CONSILIER (id\_angajat)

SELECT id\_angajat

FROM ANGAJAT

WHERE id\_angajat = 4;

UPDATE CONSILIER SET

nr\_clienti = 75

WHERE id\_angajat = 4;

INSERT INTO CONSILIER (id\_angajat)

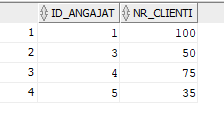
SELECT id\_angajat

FROM ANGAJAT

WHERE id\_angajat = 5;

UPDATE CONSILIER SET

nr\_clienti = 35

WHERE id\_angajat = 5;  
  


--------------------------SALARIU--------------------------

CREATE TABLE SALARIUU (

id\_salariu NUMBER(10) PRIMARY KEY,

id\_angajat NUMBER(10) NOT NULL,

CONSTRAINT fkey\_sal\_ang FOREIGN KEY (id\_angajat) REFERENCES ANGAJAT(id\_angajat) ON DELETE SET NULL,

salariu\_brut NUMBER(10),

salariu\_net VARCHAR2(60) NOT NULL,

CONSTRAINT chk\_salariu CHECK (salariu\_brut > salariu\_net)

);

INSERT INTO SALARIUU(id\_salariu, id\_angajat, salariu\_brut, salariu\_net)

VALUES (1, 0, 3500, 2800);

INSERT INTO SALARIUU(id\_salariu, id\_angajat, salariu\_brut, salariu\_net)

VALUES (2, 1, 5000, 4500);

INSERT INTO SALARIUU(id\_salariu, id\_angajat, salariu\_brut, salariu\_net)

VALUES (3, 2, 1500, 1200);

INSERT INTO SALARIUU(id\_salariu, id\_angajat, salariu\_brut, salariu\_net)

VALUES (4, 3, 4000, 3700);

INSERT INTO SALARIUU(id\_salariu, id\_angajat, salariu\_brut, salariu\_net)

VALUES (5, 4, 2000, 1800);

INSERT INTO SALARIUU(id\_salariu, id\_angajat, salariu\_brut, salariu\_net)

VALUES (6, 5, 2500, 2100);

INSERT INTO SALARIUU(id\_salariu, id\_angajat, salariu\_brut, salariu\_net)

VALUES (7, 6, 4700, 4300);

INSERT INTO SALARIUU(id\_salariu, id\_angajat, salariu\_brut, salariu\_net)

VALUES (8, 7, 1800, 1500);

INSERT INTO SALARIUU(id\_salariu, id\_angajat, salariu\_brut, salariu\_net)  
VALUES (9, 8, 1000, 970);  
  


--------------------------CONTACT--------------------------

CREATE TABLE CONTACT (

id\_contact NUMBER(10) PRIMARY KEY,

id\_angajat NUMBER(10) NOT NULL,

CONSTRAINT fkey\_contact\_ang FOREIGN KEY (id\_angajat) REFERENCES ANGAJAT(id\_angajat) ON DELETE SET NULL,

email VARCHAR(60),

telefon VARCHAR2(60) NOT NULL

);

INSERT INTO CONTACT (id\_contact, id\_angajat, email, telefon)

VALUES (0, 0, 'angajat0@banca.ro', '0368415678');

INSERT INTO CONTACT (id\_contact, id\_angajat, email, telefon)

VALUES (1, 1, 'angajat1@banca.ro', '0368213213');

INSERT INTO CONTACT (id\_contact, id\_angajat, email, telefon)

VALUES (2, 2, 'angajat2@banca.ro', '0268333444');

INSERT INTO CONTACT (id\_contact, id\_angajat, email, telefon)

VALUES (3, 3, 'angajat3@banca.ro', '0268556554');

INSERT INTO CONTACT (id\_contact, id\_angajat, email, telefon)

VALUES (4, 4, 'angajat4@banca.ro', '0368987789');

INSERT INTO CONTACT (id\_contact, id\_angajat, email, telefon)

VALUES (5, 5, 'angajat5@banca.ro', '0368223431');

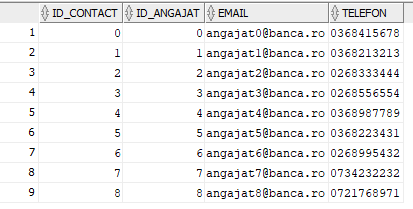
INSERT INTO CONTACT (id\_contact, id\_angajat, email, telefon)

VALUES (6, 6, 'angajat6@banca.ro', '0268995432');

INSERT INTO CONTACT (id\_contact, id\_angajat, email, telefon)

VALUES (7, 7, 'angajat7@banca.ro', '0734232232');

INSERT INTO CONTACT (id\_contact, id\_angajat, email, telefon)

VALUES (8, 8, 'angajat8@banca.ro', '0721768971');  
  


--------------------------APARTINERE--------------------------

CREATE TABLE APARTINERE (

id\_filiala NUMBER(10) CONSTRAINT pk\_fil\_ap REFERENCES FILIALA(id\_filiala),

id\_cont NUMBER(10) CONSTRAINT pk\_cont\_ap REFERENCES CONTT(id\_cont),

CONSTRAINT pk\_apartinere PRIMARY KEY(id\_filiala, id\_cont)

);

INSERT INTO APARTINERE (id\_filiala, id\_cont)

VALUES (1, 00);

INSERT INTO APARTINERE (id\_filiala, id\_cont)

VALUES (2, 01);

INSERT INTO APARTINERE (id\_filiala, id\_cont)

VALUES (2, 02);

INSERT INTO APARTINERE (id\_filiala, id\_cont)

VALUES (3, 03);

INSERT INTO APARTINERE (id\_filiala, id\_cont)

VALUES (4, 04);

INSERT INTO APARTINERE (id\_filiala, id\_cont)

VALUES (6, 05);

INSERT INTO APARTINERE (id\_filiala, id\_cont)

VALUES (1, 05);

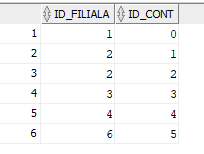
INSERT INTO APARTINERE (id\_filiala, id\_cont)

VALUES (2, 04);

INSERT INTO APARTINERE (id\_filiala, id\_cont)

VALUES (5, 00);

INSERT INTO APARTINERE (id\_filiala, id\_cont)

VALUES (4, 05);  
  


--------------------------CLIENT\_CONT--------------------------

CREATE TABLE CLIENT\_CONT (

id\_client NUMBER(10) CONSTRAINT pk\_client\_cc REFERENCES CLIENTT(id\_client),

id\_cont NUMBER(10) CONSTRAINT pk\_cont\_cc REFERENCES CONTT(id\_cont),

CONSTRAINT pk\_cc PRIMARY KEY(id\_client, id\_cont)

);

INSERT INTO CLIENT\_CONT (id\_client, id\_cont)

VALUES (1, 00);

INSERT INTO CLIENT\_CONT (id\_client, id\_cont)

VALUES (2, 01);

INSERT INTO CLIENT\_CONT (id\_client, id\_cont)

VALUES (3, 02);

INSERT INTO CLIENT\_CONT (id\_client, id\_cont)

VALUES (4, 03);

INSERT INTO CLIENT\_CONT (id\_client, id\_cont)

VALUES (5, 04);

INSERT INTO CLIENT\_CONT (id\_client, id\_cont)

VALUES (6, 05);

INSERT INTO CLIENT\_CONT (id\_client, id\_cont)

VALUES (7, 00);

INSERT INTO CLIENT\_CONT (id\_client, id\_cont)

VALUES (8, 01);

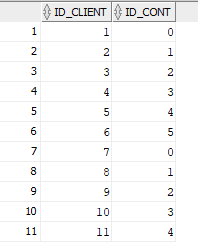
INSERT INTO CLIENT\_CONT (id\_client, id\_cont)

VALUES (9, 02);

INSERT INTO CLIENT\_CONT (id\_client, id\_cont)

VALUES (10, 03);

INSERT INTO CLIENT\_CONT (id\_client, id\_cont)

VALUES (11, 04);  
  


**11. Formulați în limbaj natural și implementați 5 cereri SQL complexe ce vor utiliza, în ansamblul lor, următoarele elemente:  
 •operație join pe cel puțin 4 tabele  
•filtrare la nivel de linii  
•subcereri sincronizate în care intervin cel puțin 3 tabele  
•subcereri nesincronizate în care intervin cel puțin 3 tabele  
•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)**

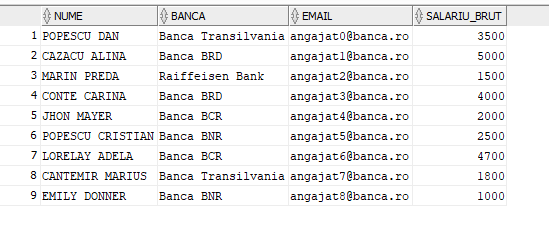
Problema 1:   
- join pe 4 tabele + filtrare la nivel de linii  
Sa se afiseze numele angajatului, numele bancii la care este acesta angajat, email-ul si salaraiul brut pe are il primeste.

SELECT a.nume, b.nume AS BANCA, c.email, s.salariu\_brut

FROM ANGAJAT a JOIN BANCA b USING (id\_banca)

JOIN CONTACT c USING (id\_angajat)

JOIN SALARIUU s USING (id\_angajat);



Problema 2:

-- clauza WITH + CASE

Sa se faca suma tranzactiilor, iar daca suma este mai mare decat 10000, sa afisam un mesaj, iar daca nu e alt mesaj.

WITH SUMA AS (SELECT SUM(totalul\_tranzactiei) TOTAL FROM TRANZACTIE)

SELECT TOTAL,

CASE

WHEN TOTAL >= 10000 THEN 'Tranzactiile au depasit totalul de 10000'

ELSE 'Tranzactiile nu au depasit totalul de 10000'

END AS "Informatii tranzactii"

FROM SUMA;  
  


Problema 3:

-- filtrare la nivel de grup, subcereri

-- Angajatii care lucreaza in banca care are numarul minim de angajati.

SELECT a.\*

FROM ANGAJAT a

WHERE a.id\_banca IN (SELECT b.id\_banca

FROM ANGAJAT a, BANCA b

WHERE a.id\_banca = b.id\_banca

GROUP BY b.nume, b.id\_banca

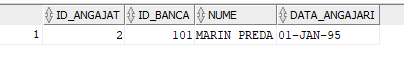
HAVING COUNT(\*) = (SELECT MIN(COUNT(\*))

FROM ANGAJAT a, BANCA b

WHERE b.id\_banca = a.id\_banca

GROUP BY b.nume, b.id\_banca)

);



Problema 4:

-- NVL

-- Sa se selecteze numele si prenumele clientului si daca clientul are cont, astfel, daca acestia au cont, se va afisa "DA"

ALTER TABLE CLIENTT

ADD are\_cont VARCHAR(255);

Select nvl(ct.id\_client, ‘NU’)

From Client c left join cont ct on c.id\_client = ct.id\_client

select nume, prenume, NVL(are\_cont,'DA') as cont\_bancar

from CLIENTT;



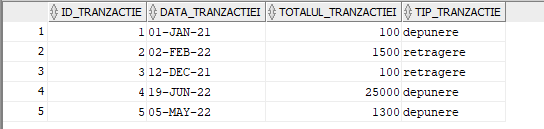
Problema 5:

-- functii pe date

-- Sa se afiseze datele despre o tranzactie care a fost facuta in mai putin de 10 luni.

SELECT id\_tranzactie, data\_tranzactiei, totalul\_tranzactiei, tip\_tranzactie

FROM TRANZACTIE

WHERE MONTHS\_BETWEEN(TO\_DATE(SYSDATE, 'DD-MM-YYYY'),data\_tranzactiei)<=10;  
  


**12.Implementarea a 3operații de actualizare sau suprimare a datelor utilizând subcereri.**

-- Adaugarea unei coloane in tabelul CLIENT

ALTER TABLE CLIENTT

ADD are\_cont VARCHAR(255);

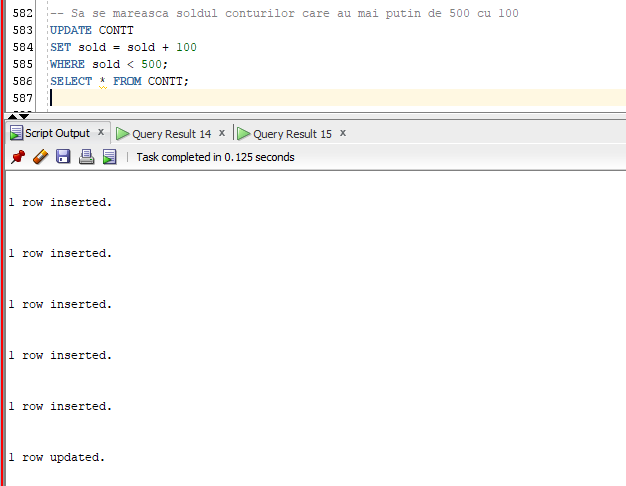
-- Sa se mareasca soldul conturilor care au mai putin de 500 cu 100

UPDATE CONTT

SET sold = sold + 100

WHERE sold < 500;

SELECT \* FROM CONTT;



-- Sa se stearga toate tranzactiile de tipul retragere

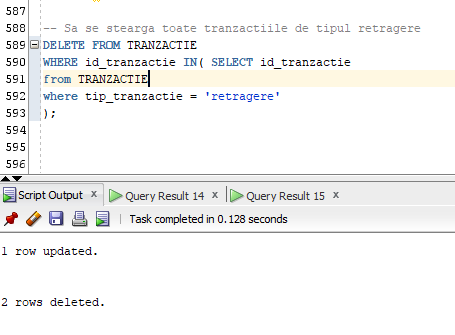
DELETE FROM TRANZACTIE

WHERE id\_tranzactie IN( SELECT id\_tranzactie

from TRANZACTIE

where tip\_tranzactie = 'retragere'

);

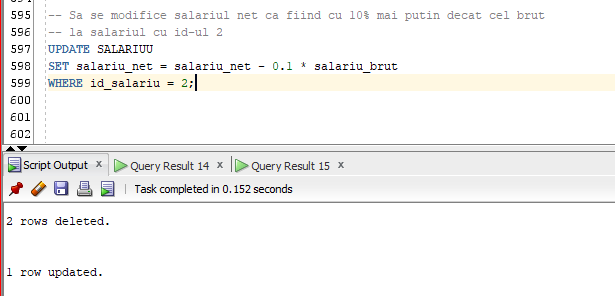


-- Sa se modifice salariul net ca fiind cu 10% mai putin decat cel brut, la salariul cu id-ul 2.

UPDATE SALARIUU

SET salariu\_net = salariu\_net - 0.1 \* salariu\_brut

WHERE id\_salariu = 2;



**13.Crearea unei secvențe ce va fi utilizatăîn inserarea înregistrărilor în tabele(punctul 10).**

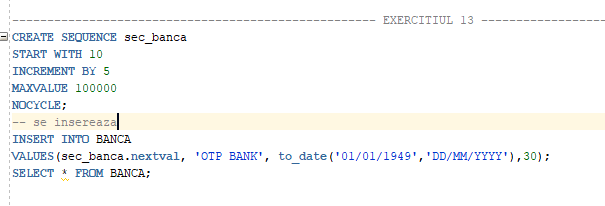
CREATE SEQUENCE sec\_banca

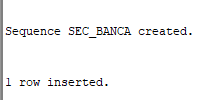
START WITH 10

INCREMENT BY 5

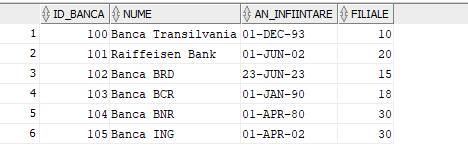
MAXVALUE 100000

NOCYCLE;

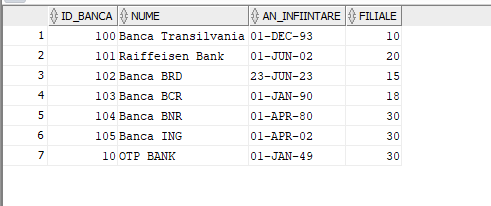




INAINTE

****

DUPA

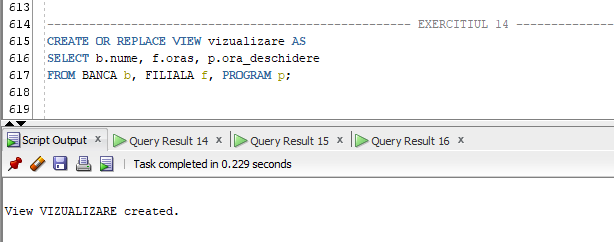


**14.Crearea unei vizualizări compuse. Dați un exemplu de operație LMD permisăpe vizualizarea respectivă și un exemplu de operație LMD nepermisă.**

CREATE OR REPLACE VIEW vizualizare AS

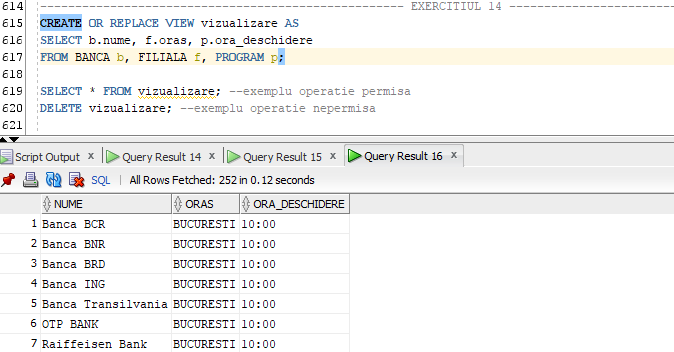
SELECT b.nume, f.oras, p.ora\_deschidere

FROM BANCA b, FILIALA f, PROGRAM p;



SELECT \* FROM vizualizare; --exemplu operatie permisa

DELETE vizualizare; --exemplu operatie nepermisa



**16. Formulați în limbaj natural și implementați în SQL: o cerere ce utilizează operația outer-joinpe minimum 4 tabele și douăcererice utilizeazăoperațiadivision.**

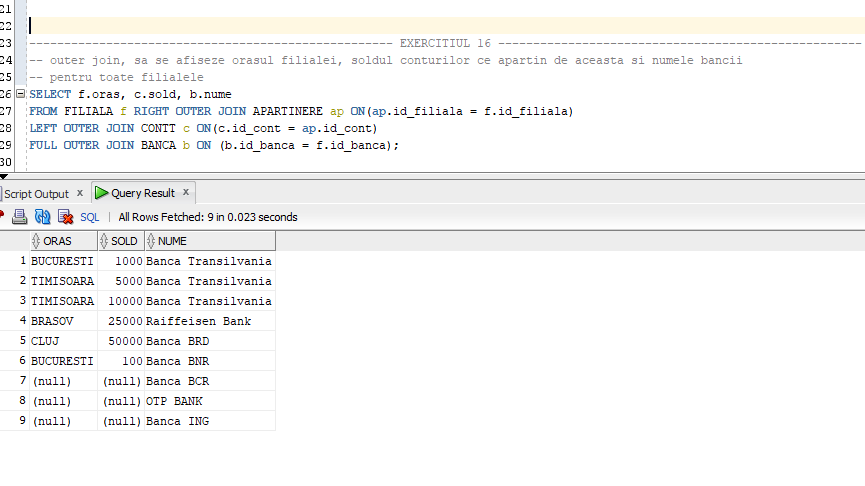
-- outer join, sa se afiseze orasul filialei, soldul conturilor ce apartin de aceasta si numele bancii, pentru toate filialele

SELECT f.oras, c.sold, b.nume

FROM FILIALA f RIGHT OUTER JOIN APARTINERE ap ON(ap.id\_filiala = f.id\_filiala)

LEFT OUTER JOIN CONTT c ON(c.id\_cont = ap.id\_cont)

FULL OUTER JOIN BANCA b ON (b.id\_banca = f.id\_banca);



-- Sa se obtina codurile imprumuturilor care au fost facute de clientului cu id-ul 11 sau 7

SELECT \* FROM IMPRUMUT;

SELECT \* FROM CLIENTT;

SELECT id\_imprumut

FROM IMPRUMUT

MINUS

SELECT id\_imprumut

FROM (SELECT id\_imprumut, id\_client

FROM (SELECT DISTINCT id\_imprumut FROM IMPRUMUT) i1,

(SELECT id\_client FROM CLIENTT WHERE id\_client = 11 or id\_client = 7) i2

MINUS

SELECT id\_imprumut, id\_client

FROM IMPRUMUT) i3;

