**Proiect Baze De Date**

**Nume:** Kayed Amar

**Grupa:** 143

**Email:** [amar.kayed@s.unibuc.ro](mailto:amar.kayed@s.unibuc.ro)

Anul Universitar 2020 - 2021

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

Exemplele din cadrul acestui proiect au ca scop proiectarea si prezentarea unui model de date ce furnizeaza informatii despre gestiunea studentilor in cadrul unei universitati, concept important in cadrul academic.

Vom prezenta modelul de date, restrictiile pe care trebuie sa le respecte si vom incerca, intr-o maniera didactica, sa construim diagrama E/R corespunzatoare, precum si diagrama conceptuala.

Modelul de date va getiona informatii legate de organizarea si functionarea diferitelor componente ale unei universitati. O universitate reprezinta un cadru academic ce contine mai multe facultati, care la randul lor contin studenti si angajati. Acest model de date este valabil pentru orice tip de universitate.

O universitate este alcatuita din mai multe facultati, fiecare facultate avand mai multe specializari. Pe langa specializari, o facultate are studenti si angajati. Studentii studiaza/invata disciplinele oferite de o specializare.

O specializare este unica in cadrul unei facultati, insa contine discipline care pot fi comune intre mai multe specializari din cadrul **aceleasi** facultati. Exista facultati diferite cu specializari sau discipline asemanatoare, dar nu identice, spre exemplu: atat FMI cat si Automatica predau materia “Baze de Date”, insa aceasta materie se diferentiaza prin cel putin specializarea si profesorul care o preda.

Angajatii lucreaza in cadrul unei facultati, acestia sunt de mai multe tipuri:

* Secretare, caz in care se ocupa cu administrarea facultatii, tot ce tine de partea organizatorica si birocratica a unei facultati.
* Profesori, caz in care se ocupa cu predarea disciplinelor si evaluarea studentilor

Un student poate urmari mai multe specializari diferite, insa nu in cadrul aceleasi facultati. Studentul poate urmari mai multe facultati diferite si in cadrul fiecarei facultati o singura specializare.

Pentru a studia in cadrul unei facultati, un student poate fi cazat la un camin. Indiferent de numarul de facultati la care studentul este inscris, acesta poate ocupa maxim un singur loc la camin(adica se poate caza doar la un singur camin).

La o disciplina participa atat profesori cat si studenti. O disciplina are implicit si o forma de examinare stabila de profesor si sustinuta de student. Forma de examinare a unei discipline poate fi de mai multe tipuri, insa in diagrama E/R am ales doar sa specificam tipul prin intermediul unui atribut, si nu prin intermediul unor subentitati, intrucat in acest context nu vom pune foarte mult accent pe diferitele tipuri de examinare. Formele de examinare sunt: examen scris, proiect, ascultare orala. Pentru toate formele de examinare trebuie sa se cunoasca nota studentului si data sustinerii.

Facultatile unei universitati ocupa implicit o locatie, aceasta locatie la randul ei apartinand unei tari.

Modelul de date este util, intrucat proiecteaza o structura ordonata in care se pot tine evidenta studentilor inscrisi la diverse facultati, precum si informatiile despre respectivele facultati si ale angajatilor din cadrul acestor facultati.

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

In acest context, pentru simplificarea notiunilor, prin specializare ne referim la un program de studiu distinct oferita de o anumita facultate si care poate fi urmarita de studenti ai facultatii respective.

Modelul de date respectă anumite restricţii de funcţionare.

* Facultatea ofera mai multe programe de studii(adica specializari) din care un student poate sa aleaga doar una singura. Studentii pot fi inscrisi la mai multe facultati, insa in cadrul unei facultati pot urma o singura speicalizare.
* In acest context, prin angajat ne referim la un salariat strict al facultatii(si prin consecinta a universitatii) si nu a oricarei alte organizatii. In acest sens, angajatii pot fi de mai multe tipuri, fiecare tip ocupand un rol esential in buna functionare a facultatii(Ex: profesor, secretara, etc).
* In cazul in care angajatul reprezinta un profesor, acesta poate preda mai multe discipline, disciplinele respective fiind urmarite de studentii inscrisi la facultate.
* Nu exista angajati care sa fie minori(care sa nu aibe varsta de cel putin 18 ani).
* Facultatea are implicit o adresa, care se afla intr-o locatie. Aceasta locatie automat se afla intr-o tara si intr-o regiune.
* Pentru modelul de date s-a considerat faptul ca toate facultatile apartin aceleasi universitati, de aici motivandu-se absenta unei entitati pentru universitati.
* Cu toate ca toate facultatile apartin aceleasi universitati, acestea pot prezenta specializari/discipline asemanatoare iar un student poate fi inscris la mai multe facultati.
* Pentru fiecare disciplina s-a considerat obligatoriu o forma de examinare.
* Toti studentii inscrisi la o facultate trebuie sa urmeze obligatoriu o specializare din cadrul facultatii respective.
* Pentru orice disciplina este obligatoriu ca numarul de ore predate sa fie de cel putin 1, adica sa nu existe discipline care nu sunt predate/parcurse.

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

Pentru modelul de date referitor la gestiunea unei universitati, structurile ANGAJAT, SECRETARA, PROFESOR, DISCIPLINA, EXAMINARE, STUDENT, CAMIN, SPECIALIZARE, FACULTATE, LOCATIE, TARA reprezinta entitati.

Vom prezenta entităţile modelului de date, dând o descriere completă a fiecăreia. De asemenea, pentru fiecare entitate se va preciza cheia primară.

Toate entitatile care fi prezentate sunt independente, cu exceptia entitatii EXAMINARE, aceasta depinzand de entitatea DISCIPLINA(nu poate exista examinare daca nu exista disciplina/materie pentru acest examen).

1. ANGAJAT = persoana fizica/salariat in cadrul unei facultati si, implicit, in cadrul unei universitati. Angajatul se ocupa cu organizarea si gestionarea diferitelor componente care intra buna functionare a unei facultati. Aceasta entitate reprezinta o superclasa pentru entitatile *PROFESOR* si *SECRETARA*, intrucat un angajat in cadrul unei facultati poate fi un profesor sau o secretara, sau niciuna. Cheia primara a acestei superentitati este *cod\_angajat*.
2. SECRETARA = subentitate a entitatii ANGAJAT, aceasta reprezinta reprezinta o persoana fizica care se ocupa cu administrarea unei facultati, cu organizarea si intretinerea activitatilor academice precum si a componentelor birocratice/juridice. Fiind o subentitate, aceasta mosteneste cheia primara a superclasei ANGAJAT, anume atributul *cod\_angajat*.
3. PROFESOR = subentitate a entitatii ANGAJAT, aceasta reprezinta un angajat care se ocupa cu predarea disciplinelor din cadrul unei specializari a facultatii, precum si de evaluarea studentilor inscrisi la facultatea respectiva. Fiind o subentitate, aceasta mosteneste cheia primara a superclasei ANGAJAT, anume atributul *cod\_angajat*.
4. DISCIPLINA = concept academic in cadrul unei facultati, aceasta reprezinta un set de competente menite a fi predate studentilor, urmand ca acestia sa asimilize cunostiinta si sa fie evaluati din respectivele competente. Cheia primara a entitatii este *cod\_disciplina*.
5. EXAMINARE = activitate de evaluare a competentelor dobandite de student in cadrul unei discipline. Aceasta entitate este dependenta de enitatea DISCIPLINA, care contine informatii despre materia pentru care se face examinarea. Cheia primara a entitatii este compusa din *cod\_examinare* si *cod\_disciplina*.
6. STUDENT = persoana fizica, inscrisa in cadrul unei facultati si care urmeaza o specializare a facultatii respective. Aceasta entitate intra in relatie cu disciplinele predate in cadrul facultatii/specializarii, precum si cu profesorul ce preda respectiva disciplina. Cheia primara a acestei entitati este *cod\_student*(initial voiam sa pun numar\_matricol, intrucat acest atribut este unic in cadrul unei universitati, sau cnp).
7. CAMIN = persoana juridica, responsabila cu gazduirea studentilor din cadrul unei facultati. Cheia primara a entitatii este *cod\_camin*.
8. SPECIALIZARE = entitate care defineste programul de studiu din cadrul unei facultati. Aceasta entitate contine informatii legate de domeniul curricular al unei specializari din cadrul unei facultati. Cheia primara a acestei entitati este *cod\_specializare.*
9. FACULTATE = persoana juridica, reprezentand o entitate institutionala responsabila pentru facilitarea unui cadru academic formal. Aceasta entitate contine informatii legate de datele unei facultati si intra in relatii angajati si studenti. Cheia primara a acestei entitati este *cod\_facultate.*
10. LOCATIE = adresa fizica, aceasta entitate contine informatii legate de locatia fizica a unei facultati. Cheia primara a acestei entitati este *cod\_locatie.*
11. TARA = teritoriu geografic care inglobeaza mai multe locatii. Cheia primara a acestei entitati este *cod\_tara.*

**4. Descrierea relațiilor, incluzând precizarea cardinalității acestora.**

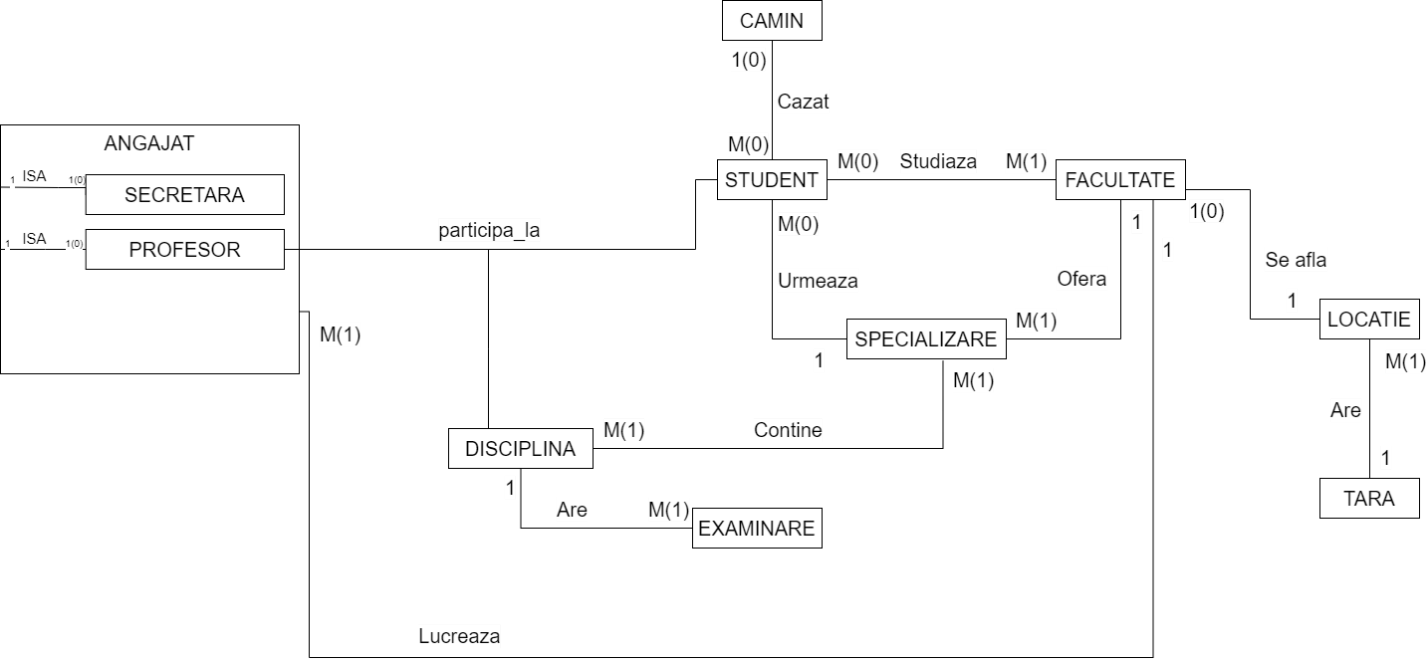
Vom prezenta relaţiile modelului de date, dând o descriere completă a fiecăreia. De fapt, denumirile acestor legături sunt sugestive, reflectând conţinutul acestora şi entităţile pe care le leagă. Pentru fiecare relaţie se va preciza cardinalitatea minimă şi maximă.

* ANGAJAT ISA SECRETARA = relatie de mostenire care leaga superclasa ANGAJAT si subclasa SECRETARA, reflectand legatura dintre acestea(o secretara este un angajat). Ea are cardinalitatea **minima 1:0**(one-to-zero), intrucat o secretara **trebuie sa fie si este** un angajat prin definitie, insa un angajat **nu trebuie** sa fie neaparat o secretara(poate fi si profesor sau orice altceva). De asemenea, relatia are cardinalitatea **maxima 1:1**(one-to-one), intrucat o secretara **poate fi maxim** un singur angajat(nu poti fi o singura secretara si sa ocupi pozitiile a mai multor angajati) si un angajat **poate fi maxim** o singura secretara(nu mai multe concomitent).
* ANGAJAT ISA PROFESOR = relatie de mostenire care leaga superclasa ANGAJAT si subclasa PROFESOR, reflectand legatura dintre acestea(un profesor este un angajat). Ea are cardinalitatea **minima 1:0**(one-to-zero), intrucat un profesor **trebuie sa fie si este** un angajat prin definitie, insa un angajat **nu trebuie** sa fie neaparat un profesor(poate fi si secretara sau orice altceva). De asemenea, relatia are cardinalitatea **maxima 1:1**(one-to-one), intrucat un profesor **poate fi maxim** un singur angajat(nu poti fi un singur profesor si sa ocupi pozitiile a mai multor angajati) si un angajat **poate fi maxim** un singur profesor(nu mai multi concomitent).
* ANGAJAT lucreaza FACULTATE = relatie care leaga superentitatea ANGAJAT si entitatea FACULTATE, reflectand legatura dintre acestea(la ce facultate lucreaza un angajat). Ea are cardinalitatea **minima 1:1**(one-to-one), intrucat un angajat **trebuie** sa lucreze la minim o facultate(altfel nu ar mai fi angajat in contextul universitatii) si o facultate **trebuie** sa contina minim un angajat(altfel s-ar desfiinta facultatea). De asemenea, aceasta relatie are cardinalitatea **maxima 1:n**(one-to-many), intrucat o facultate **poate** avea mai multi angajati in acelasi timp(mai multi profesori si mai multe secretare), insa un angajat **poate** lucra doar la o singura facultate si nu mai multe.
* STUDENT cazat CAMIN = relatie care leaga entitatile STUDENT si CAMIN, reflectand legatura dintre acestea(la ce camin este cazat un student). Ea are cardinalitatea **minima 0:0**(zero-to-zero), intrucat un student **nu trebuie** sa fie cazat neaparat la un camin, iar un camin **nu trebuie** sa aibe neaparat studenti in el(poate sa fie si gol). De asemenea, relatia are cardinalitatea **maxima 1:n**(one-to-many), intrucat un camin **poate** gazdui mai multi studenti, insa un student **poate** fi cazat la maxim un singur camin.
* STUDENT studiaza FACULTATE = relatie care leaga entitatile STUDENT si FACULTATE, reflectand legatura dintre acestea(la ce facultate studiaza un student). Ea are cardinalitatea **minima 1:0**(one-to-zero), intrucat un student **trebuie** sa fie inscris si sa studieze la minim o facultate(altfel nu ar mai fi student), insa o facultate **nu trebuie** neaparat sa aibe vreun student. De asemenea, aceasta relatie are cardinalitatea **maxima n:n**(many-to-many), intrucat un student **poate** fi inscris si sa studieze la mai multe facultati(in viata reala de obicei doar 2 facultati dar optiunea de a urma 3 sau mai multe este posibila), in timp ce o facultate **poate** avea mai multi studenti. Aceasta relatie many-to-many va crea un nou tabel in diagrama conceptuala, tabel care va avea o cheie primara compusa din cheile primare ale celor doua tabele aflate in relatie.
* STUDENT urmeaza SPECIALIZARE = relatie care leaga entitatile STUDENT si SPECIALIZARE, reflectand legatura dintre acestea(ce specializare urmeaza un student). Ea are cardinalitatea **minima 1:0**(one-to-zero), intrucat in cadrul unei facultati un student **trebuie** sa urmeze minim o specializare, insa o specializare **nu trebuie** sa fie urmarita de minim un student. De asemenea, relatia are cardinalitatea **maxima 1:n**(one-to-many), intrucat o specializare **poate** fi urmarita de mai multi studenti in acelasi timp(exista mai multi studenti care urmeaza specializarea de informatica spre exemplu), insa un student **poate** urmari maxim o singura specializare in cadrul unei singure facultati, acesta putand studia in mai multe facultati si pentru fiecare dintre aceste facultati poate urmari o singura specializare.
* FACULTATE ofera SPECIALIZARE = relatie care leaga entitatile FACULTATE si SPECIALIZARE, reflectand legatura dintre acestea(ce specializari ofera o facultate). Ea are cardinalitatea **minima 1:1**(one-to-one), intrucat o facultate **trebuie** sa ofere minim o specializare(adica un program de studiu), iar o specializare **trebuie** sa fie oferita(adica sa apartina) unei singure facultati(nu exista specializari care sa nu fie in cadrul unei facultati). De asemenea, relatia are cardinalitatea **maxima 1:n**(one-to-many), intrucat o facultate **poate** oferi mai multe specializari(cum ar fi FMI-ul, oferind specializare de mate, info si CTI), iar o specializare **poate** fi oferita de o singura facultate(exista, intr-adevar, aceeasi denumire pentru o specializare cum ar fi CTI de la FMI si CTI de la automatica, insa acestea nu sunt exact aceeasi specializare, au materii comune dar si materii diferite, deci nu sunt identice, acestea fiind diferite prin cel putin cheia primara care este unica fiecarei entitati).
* FACULTATE se afla LOCATIE = relatie care leaga entitatile FACULTATE si SPECIALIZARE, reflectand legatura dintre acestea(in ce locatie se afla o facultate). Ea are cardinalitatea **minima 1:0**(one-to-zero), intrucat o facultate **trebuie** sa se afle intr-o locatie, in timp ce intr-o locatie **nu trebuie** sa se afle neaparat o facultate(nu exista orase in care toate cladirile sunt doar facultati). De asemenea, relatia are cardinalitatea **maxima 1:1**(one-to-one), intrucat o facultate se **poate** afla maxim intr-o singura locatie(in acest context facultatile nu au mai multe cladiri separate), in timp ce o locatie **poate** avea maxim o singura facultate.
* LOCATIE are TARA = relatie care leaga entitatile LOCATIE si TARA, reflectand legatura dintre acestea(in ce tara se afla o anumita locatie). Ea are cardinalitatea **minima 1:1**(one-to-one), intrucat o locatie **trebuie** sa se afle intr-o tara, iar o tara **trebuie** sa contina minim o locatie(altfel nu ar mai fi tara si ar fi doar un teritoriu pustiu nelocuit). De asemenea, relatia are cardinalitatea **maxima 1:n**(one-to-many), intrucat o tara **poate** avea mai multe locatii, in timp ce o locatie **poate** sa apartina maxim unei singure tari.
* SPECIALIZARE contine DISCIPLINA = relatie care leaga entitatile SPECIALIZARE si DISCIPLINA, reflectand legatura dintre acestea(ce discipline se studiaza in cadrul unei specializari al unei facultati). Ea are cardinalitatea **minima 1:1**(one-to-one), intrucat o specializare **trebuie** sa contina minimum o disciplina(altfel nu ar mai fi specializare), in timp ce o disciplina **trebuie** sa apartina minim unei specializari. De asemenea, relatia are cardinalitatea **maxima n:n**(many-to-many), intrucat o specializare **poate** sa aibe mai multe discipline, in timp ce aceeasi disciplina **poate** sa apartina mai multor specializari(cum ar fi exemplul de la FMI pentru materia “Baze de Date”, care apartine atat specializarii de informatica, cat si pentru studentii de la specializarea matematica-informatica). Aceasta relatie many-to-many va crea un nou tabel in diagrama conceptuala, tabel care va avea o cheie primara compusa din cheile primare ale celor doua tabele aflate in relatie.
* DISCIPLINA are EXAMINARE = relatie care leaga entitatile DISCIPLINA si EXAMINARE, reflectand legatura dintre acestea(ce forma de examinare are o anumita disciplina). Ea are cardinalitatea **minima 1:1**(one-to-one), intrucat o disciplina **trebuie** sa aibe o forma de examinare(fie ea examen, proiect sau ascultare), in timp ce o examinare **trebuie** sa apartina unei discipline. De asemenea, relatia are cardinalitatea **maxima 1:n**(one-to-many), intrucat o disciplina **poate** avea mai multe examinari(mai multe teste de laborator+proiecte, etc), in timp ce o examinare **poate** sa apartina maxim unei singure discipline(nu se poate da un singur examen pentru doua sau mai multe materii simultan)
* STUDENT participa\_la DISCIPLINA predata\_de PROFESOR = relatie de tip 3 ce leaga entitatile STUDENT, DISCIPLINA si PROFESOR, reflectand ce student participa la disciplina predata de un anumit profesor. Din moment ce este o relatie de tip 3, aceasta se va transforma intr-un tabel care va avea o cheie primara compusa din cele 3 chei primare ale tabelelor aflate in relatie, acest lucru se poate observa si in diagrama conceptuala, denumirea acestui nou tabel nou fiind “PARTICIPA”, intrucat profesorul preda disciplina la care **participa** un student. Datorita acestei relatii de tip 3, acum putem raspunde la intrebarea “la ce disciplina participa un student pentru un anumit profesor”, intrucat o relatie de tip 3 este diferita de 3 relatii de tip 2.

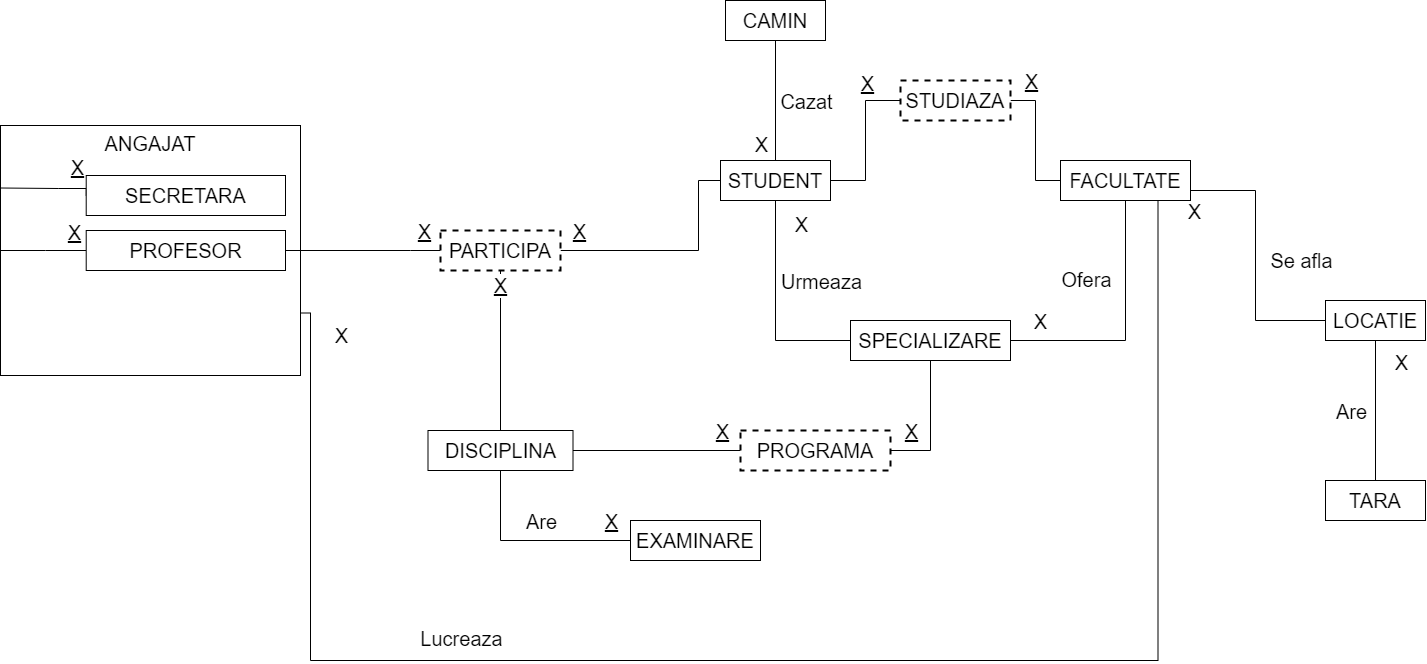
**5. Descrierea atributelor, incluzând tipul de date și eventualele constrângeri, valori implicite, valori posibile ale atributelor.**

1. Entitatea independenta ANGAJAT are ca atribute:
   * ***cod\_angajat*** = variabila de tip intreg, de lungime maxima 5, care reprezinta codul unui angajat, aceasta fiind o cheie primara. Prin urmare, acest atribut nu poate fi NULL. (Ex: 12, 1234, 51)
   * ***nume*** = variabila de tip caracter, de lungime maxima 25, care reprezinta numele angajatului. Acest atribut nu poate fi NULL. (Ex: ‘Popescu’, ‘Mihailescu’)
   * ***prenume*** = variabila de tip caracter, de lungime maxima 25, care reprezinta prenumele angajatului. Acest atribut nu poate fi NULL. (Ex: ‘Mihai’, ‘Andrei’)
   * ***data\_nastere*** = variabila de tip data calendaristica, care reprezinta data nasterii angajatului respectiv. Acest atribut va avea ca valoare implicita data curenta(folosind SYSDATE-ul din SQL). Acest atribut are constrangere de a avea valori posibile care sa fie cu minim 18 ani inainte de data curenta, adica nu exista angajati minori. Acest atribut poate fi si NULL. (Ex: ’17-JUN-87’)
   * ***salariu*** = variabila de tip real, cu 10 cifre in total, dintre care 2 sunt cifre zecimale, care reprezinta salariul lunar al unui angajat.
   * ***cod\_facultate*** = variabila de tip intreg, de lungime maxima 5, care reprezinta codul facultatii la care lucreaza angajatul. Atributul reprezinta o cheie externa si trebuie sa corespunda unei valori a cheii primare din tabelul FACULTATE. In acest context, un angajat trebuie sa lucreze la facultate, deci acest atribut nu poate fi null. (Ex: 120, 12340, 510)
2. Subentitatea SECRETARA mosteneste toate atributele superclasei ANGAJAT prin intermediul cheii primare comune dintre aceste doua tabele. Enitatea SECRETARA este dependenta de entitatea ANGAJAT. Prin urmare, subentitatea SECRETARA are atributele:
   * ***cod\_angajat*** = variabila de tip intreg, de lungime maxima 5, care reprezinta codul unui angajat, aceasta fiind o cheie primara in tabelul SECRETARA. Cheia primara din SECRETARA poate fi folosita pentru a accesa atributele corespunzatoare din tabelul ANGAJAT. Cheia primara din SECRETARA trebuie sa corespunda unei chei primare din ANGAJAT, ambele neputand fii NULL. (Ex: 12, 1234, 51)
   * ***tehnologie\_favorita*** = variabila de tip caracter, de lungime maxima 25, care reprezinta tehnologia favorita utilizata de secretara pentru a administra facultatea la care este angajata. Aceast atribut poate fi NULL. (Ex: Word, Powerpoint, Excel, etc)
   * ***specializare*** = variabila de tip caracter, de lungime maxima 25, care reprezinta specializarea secretarei. Aceast atribut poate fi NULL. (Ex: ‘Organizare’, ‘Administrare’, ‘Statistica’)
3. Subentitatea PROFESOR mosteneste toate atributele superclasei ANGAJAT, prin intermediul cheii primare comune dintre aceste doua tabele. Entitatea PROFESOR este dependenta de entitatea ANGAJAT. Prin urmare, subeintatea PROFESOR are atributele:
   * ***cod\_angajat*** = variabila de tip intreg, de lungime maxima 5, care reprezinta codul unui angajat, aceasta fiind o cheie primara in tabelul PROFESOR. Cheia primara din PROFESOR poate fi folosita pentru a accesa atributele corespunzatoare din tabelul ANGAJAT. Cheia primara din PROFESOR trebuie sa corespunda unei chei primare din ANGAJAT, ambele neputand fii NULL. (Ex: 12, 1234, 51)
   * ***nota\_titularizare*** = variabila de tip numar real, cu 4 cifre, dintre care 2 sunt zecimale, care reprezinta nota obtinuta de profesor in examenul de titularizare. Acest atribut poate fi NULL, intrucat in acest context pot exista profesori fara titularizare. De asemenea, nota maxima este 10.00 (Ex: 9.54, 6.21, 10.00)
   * ***tip\_profesor*** = variabila de tip caracter, de lungime maxima 15, care reprezinta tipul profesorului. Valorile posibile sunt: ‘Cursant’, ‘Seminarist‘, ‘Laborant’. Acest atribut poate fi NULL, caz in care se subintelege ca profesorul este invitat special pentru a tine o ora, iar acesta nu se incadreaza in categoriile anterior mentionate.
4. Entitea independenta DISCIPLINA are ca atribute:
   * ***cod\_disciplina*** = variabila de tip intreg, de lungime maxima 5, care reprezinta codul disciplinei, aceasta fiind o cheia primara pentru DISCIPLINA. Prin urmare, acest atribut nu poate fi NULL. (Ex: 10, 20, 30)
   * ***denumire*** = variabila de tip caracter, de lungima maxima 50, care reprezinta denumirea disciplinei. Acest atribut nu poate fi NULL. (Ex: ‘Baze de Date’, ‘Programare Web’)
   * ***nr\_ore*** = variabila de tip intreg, de lungime maxima 3, care reprezinta numarul total de ore predate pentru aceasta disciplina intr-un interval de 2 saptamani. Acest atribut nu poate fi NULL. (Ex: 3, caz in care se face un curs saptamanal si un alt curs odata la doua saptamani)
5. Entitatea dependenta EXAMINARE are ca atribute:
   * ***cod\_examinare*** = variabila de tip intreg, de lungime maxima 5, care reprezinta codul examinarii, aceasta facand parte din cheia primara compusa pentru EXAMINARE. Prin urmare, acest atribut nu poate fi NULL. (Ex: 10, 20, 30)
   * ***cod\_disciplina*** = variabila de tip intreg, de lungime maxima 5, care reprezinta codul disciplinei(refera cheia primara din DISCIPLINA), aceasta fiind a doua jumatate din cheia primara compusa pentru EXAMINARE. Prin urmare, acest atribut nu poate fi NULL. (Ex: 100, 200, 300). **Cheia primara a tabelului EXAMINARE este compusa din cod\_examinare si cod\_disciplina.**
   * ***forma*** = variabila de tip caracter, de lungime maxima 25, care reprezinta tipul/forma de examinare. Aceast atribut poate fi NULL. (Ex: ‘Examen Scris’, ‘Proiect’)
   * ***nota*** = variabila de tip numar real, de lungime maxima 4, dintre care 2 cifre sunt zecimale, care reprezinta nota obtinuta de student la examinare. Aceast atribut trebuie sa fie mai mic sau egal cu 10 si poate fi NULL.
6. Entitatea independenta STUDENT are ca atribute:
   * ***cod\_student*** = variabila de tip intreg, de lungime maxima 5, care reprezinta codul unic al unui student, acest atribut reprezinta o cheie primara si putea fi denumit si “nr\_matricol” sau “cnp”. Prin urmare, acest atribut nu poate fi NULL. (Ex: 10, 20, 30)
   * ***nume*** = variabila de tip caracter, de lungime maxima 25, care reprezinta numele studentului. Acest atribut nu poate fi NULL. (Ex: ‘Popescu’, ‘Mihailescu’)
   * ***prenume*** = variabila de tip caracter, de lungime maxima 25, care reprezinta prenumele studentului. Acest atribut nu poate fi NULL. (Ex: ‘Mihai’, ‘Andrei’)
   * ***data\_nastere*** = variabila de tip data calendaristica, care reprezinta data nasterii studentului respectiv. (Ex: ’17-JUN-87’)
   * ***cod\_specializare*** = variabila de tip intreg, de lungime maxima 5, care reprezinta codul specializarii pe care studentul o urmeaza. Acest atribut reprezinta o cheie externa care refera cheia primara din SPECIALIZARE, deci poate fi si NULL.
   * ***cod\_camin*** = variabila de tip intreg, de lungime maxima 5, care reprezinta codul caminului la care este cazat studentul, acest atribut reprezinta o cheie externa care refera cheia primara din tabelul CAMIN. Prin urmare, acest atribut poate fi NULL. (Ex: 10, 20, 30)
7. Entitatea independenta CAMIN are ca atribute:
   * ***cod\_camin*** = variabila de tip intreg, de lungime maxima 5, care reprezinta codul unic al unui camin, acest atribut reprezinta o cheie primara. Prin urmare, acest atribut nu poate fi NULL. (Ex: 10, 20, 30)
   * ***denumire*** = variabila de tip caracter, de lungime maxima 30, care reprezinta numele caminului. Acest atribut poate fi NULL. (Ex: ‘Grozavesti’, ‘Regie’)
   * ***nr\_camere*** = variabila de tip intreg, de lungime maxima 4, care reprezinta numarul total de camere pe care caminul il are. Acest atribut poate fi si NULL.(Ex: 1024, 10, 502)
8. Entitatea independenta SPECIALIZARE are ca atribute:
   * ***cod\_specializare*** = variabila de tip intreg, de lungime maxima 5, care reprezinta codul specializarii. Acest atribut reprezinta o cheie primara, deci nu poate fi NULL.
   * ***denumire*** = variabila de tip caracter, de lungime maxima 30, care reprezinta denumirea specializarii. Acest atribut nu poate fi NULL.
   * ***ani*** = variabila de tip intreg, de lungime maxima 1, care reprezinta numarul de ani necesari pentru a termina specializarea. Acest atribut nu poate fi NULL.
   * ***cod\_facultate*** = variabila de tip intreg, de lungime maxima 5, care reprezinta codul facultatii care prezinta aceasta specializare. Acest atribut reprezinta o cheie externa, deci poate fi NULL.
9. Entitatea independenta FACULTATE are ca atribute:
   * ***cod\_facultate*** = variabila de tip intreg, de lungime maxima 5, care reprezinta codul facultatii. Acest atribut reprezinta o cheie primara, deci nu poate fi NULL.
   * ***denumire*** = variabila de tip caracter, de lungime maxima 50, care reprezinta denumirea facultatii. Acest atribut nu poate fi NULL.
   * ***ranking*** = variabila de tip intreg, de lungime maxima 4, care reprezinta ranking-ul facultatii respective in top-ul tuturor facultatilor. Acest atribut poate fi NULL.
10. Entitatea independenta LOCATIE are ca atribute:
    * ***cod\_locatie*** = variabila de tip intreg, de lungime maxima 5, care reprezinta codul locatiei. Acest atribut reprezinta o cheie primara, deci nu poate fi NULL.
    * ***strada*** = variabila de tip caracter, de lungime maxima 25, care reprezinta numele strazii. Acest atribut poate fi NULL.
    * ***oras*** = variabila de tip caracter, de lungime maxima 25, care reprezinta numele orasului. Acest atribut poate fi NULL.
    * ***cod\_tara*** = variabila de tip intreg, de lungime maxima 5, care reprezinta codul tarii in care se afla locatia respectiva. Acest atribut reprezinta o cheie externa care refera cheia primara din TARA.
11. Entitatea independenta TARA are ca atribute:
    * ***cod\_tara*** = variabila de tip intreg, de lungime maxima 5, care reprezinta codul tari. Acest atribut reprezinta o cheie primara, deci nu poate fi NULL.
    * ***nume*** = variabila de tip caracter, de lungime maxima 25, care reprezinta numele tarii. Acest atribut poate fi NULL.
    * ***continent*** = variabila de tip caracter, de lungime maxima 25, care reprezinta numele continentului in care se afla tara. Acest atribut poate fi NULL.
12. Relatia STUDENT participa\_la DISCIPLINA predata\_de PROFESOR are ca atribute: cod\_angajat, cod\_disciplina, cod\_student. Aceste atribute trebuie sa refere cheile primare din PROFESOR, DISCIPLINA, STUDENT. Aceasta relatie formeaza tabelul asociativ PARTICIPA. De asemenea, atributele formeaza o cheie primara compusa pentru tabelul relatiei
13. Relatia STUDENT studiaza FACULTATE are ca atribute: cod\_student, cod\_facultate. Aceste atribute refera cheile primare din tabelele STUDENT si FACULTATE. Aceste atribute formeaza o cheie primara compusa pentru tabelul asociativ STUDIAZA.
14. Relatia SPECIALIZARE contine DISCIPLINA are ca atribute: cod\_disciplina, cod\_specializare. Aceste atribute refera cheile primare din tabelele SPECIALIZARE si DISCIPLINA. Aceste atribute formeaza o cheie primara compusa pentru tabelul asociativ PROGRAMA.

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

****

**7. Realizarea diagramei conceptuale corespunzătoare diagramei entitate-relație proiectata 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 proiectata la punctul 7.**

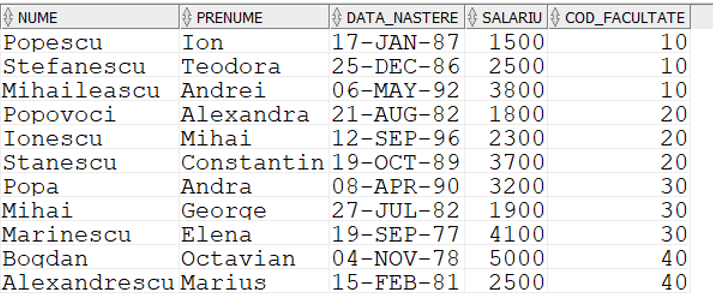
Scheme relationale:

* ANGAJAT(cod\_angajat#, nume, prenume, data\_nastere, salariu, cod\_facultate)
* SECRETARA(cod\_angajat#, tehnologie\_favorita, specializare)
* PROFESOR(cod\_angajat#, nota\_titularizare, tip\_profesor)
* DISCIPLINA(cod\_disciplina#, denumire, nr\_ore)
* EXAMINARE(cod\_examinare#, cod\_disciplina#, forma, nota)
* STUDENT(cod\_student#, nume, prenume, data\_nastere, cod\_specializare, cod\_camin)
* CAMIN(cod\_camin#, denumire, nr\_camere)
* SPECIALIZARE(cod\_specializare#, denumire, ani, cod\_facultate)
* FACULTATE(cod\_facultate#, denumire, ranking)
* LOCATIE(cod\_locatie#, strada, oras, cod\_tara)
* TARA(cod\_tara#, nume, continent)
* PARTICIPA(cod\_angajat#, cod\_disciplina#, cod\_student#)
* STUDIAZA(cod\_student#, cod\_facultate#)
* PROGRAMA(cod\_disciplina#, cod\_specializare#)

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

Vom ilustra normalizarea folosindu-ne de entitatea **Angajat**

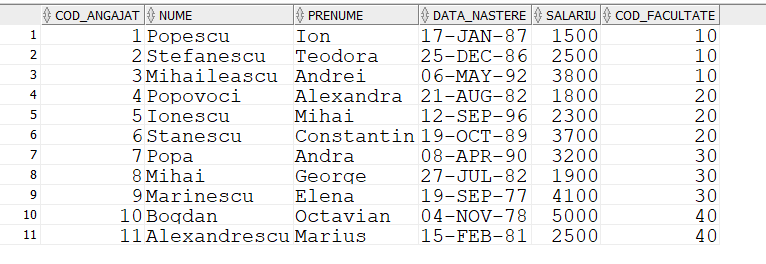
Tabelul angajat:



**FN1**

Pentru a realiza normalizarea la FN1, trebuie sa tinem cont ca entitatea noastra sa respecte urmatoarele proprietati:

* Fiecare celula din tabel trebuie sa contina o singura valoare;
* Liniile din tabel trebuie sa aiba un identificator unic (pentru asta vom adauga cod\_angajat).



Dupa cum se observa din poza anterioara, fiecare celula contine o singura valoare, iar toate liniile tabelului sunt diferentiate prin atributul unic(cheia primara) “cod\_angajat”.

**FN2**

Pentru a realiza normalizarea la FN2, trebuie sa avem in vedere faptul ca toate atributele care nu reprezinta chei sa fie dependente de chei.

Observam faptul ca putem sa identificam id-ul campionatului, folosindu-ne doar de numele echipelor, deoarece astfel putem sa determinam tara din care provin, implicit constatand si campionatul in care isi disputa meciurile. Asemenea, putem sa deducem id-ul competitiei si data\_calificarii, folosindu-ne doar de id-ul calificatei. In final, vom obtine 3 tabele.

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

CREATE TABLE angajat(

cod\_angajat NUMBER(5) CONSTRAINT pk\_angajat PRIMARY KEY,

nume VARCHAR2(25) CONSTRAINT nume\_angajat NOT NULL,

prenume VARCHAR2(25) CONSTRAINT prenume\_angajat NOT NULL,

data\_nastere DATE default add\_months(sysdate, -12\*18),

salariu NUMBER(10,2),

cod\_facultate NUMBER(5)

/\*

Aici practic am incercat sa setez data de nastere default ca fiind ziua de azi - 18 ani(varsta minima pentru angajati este de 18 ani)

Insa am aflat din documentatie ca nu se pot folosi functii sau sysdate in clauza check.

Cum se poate adauga o constrangere pentru a limita intervalul de valori pentru o data in SQL?

CONSTRAINT data\_nastere\_ang\_valida CHECK(data\_nastere <= add\_months(sysdate, -12\*18))

\*/

);

CREATE TABLE facultate(

cod\_facultate NUMBER(5) CONSTRAINT pk\_facultate PRIMARY KEY,

denumire VARCHAR2(50) CONSTRAINT denumire\_facultate NOT NULL,

ranking NUMBER(4)

);

ALTER TABLE angajat

ADD CONSTRAINT fk\_angajat\_facultate FOREIGN KEY(cod\_facultate) REFERENCES facultate(cod\_facultate);

CREATE TABLE secretara(

cod\_angajat NUMBER(5) CONSTRAINT pk\_secretara PRIMARY KEY,

tehnologie\_favorita VARCHAR2(25),

specializare VARCHAR2(25),

CONSTRAINT pk\_secretara\_valid FOREIGN KEY(cod\_angajat) REFERENCES angajat(cod\_angajat)

);

CREATE TABLE profesor(

cod\_angajat NUMBER(5) CONSTRAINT pk\_profesor PRIMARY KEY,

nota\_titularizare NUMBER(4,2),

tip\_profesor VARCHAR2(15),

CONSTRAINT pk\_profesor\_valid FOREIGN KEY(cod\_angajat) REFERENCES angajat(cod\_angajat)

);

CREATE TABLE disciplina(

cod\_disciplina NUMBER(5) CONSTRAINT pk\_disciplina PRIMARY KEY,

denumire VARCHAR2(50) CONSTRAINT denumire\_disciplina NOT NULL,

nr\_ore NUMBER(3) CONSTRAINT nr\_ore\_disciplina NOT NULL

);

CREATE TABLE examinare(

cod\_examinare NUMBER(5),

cod\_disciplina NUMBER(5) CONSTRAINT fk\_examinare\_disciplina REFERENCES disciplina(cod\_disciplina),

forma VARCHAR2(25),

nota NUMBER(4,2),

CONSTRAINT pk\_examinare PRIMARY KEY(cod\_examinare, cod\_disciplina)

);

CREATE TABLE camin(

cod\_camin number(5) constraint pk\_camin primary key,

denumire varchar2(30),

nr\_camere number(4)

);

CREATE TABLE specializare(

cod\_specializare number(5) constraint pk\_specializare primary key,

denumire varchar2(30) constraint denumire\_specializare not null,

ani number(1) constraint ani\_specializare not null,

cod\_facultate number(5) constraint fk\_specializare\_facultate references facultate(cod\_facultate)

);

CREATE TABLE student(

cod\_student number(5) constraint pk\_student primary key,

nume varchar2(25) constraint nume\_student not null,

prenume varchar2(25) constraint prenume\_student not null,

data\_nastere date,

cod\_specializare number(5) constraint fk\_student\_specializare references specializare(cod\_specializare),

cod\_camin number(5) constraint fk\_student\_camin references camin(cod\_camin)

);

CREATE TABLE locatie(

cod\_locatie number(5) constraint pk\_locatie primary key,

strada varchar2(25),

oras varchar2(25),

cod\_tara number(5)

);

CREATE TABLE tara(

cod\_tara number(5) constraint pk\_tara primary key,

nume varchar2(25),

continent varchar2(25)

);

ALTER TABLE locatie

ADD CONSTRAINT fk\_locatie\_tara FOREIGN KEY(cod\_tara) REFERENCES tara(cod\_tara);

CREATE TABLE participa(

cod\_angajat number(5) constraint fk\_participa\_angajat references angajat(cod\_angajat),

cod\_disciplina number(5) constraint fk\_participa\_disciplina references disciplina(cod\_disciplina),

cod\_student number(5) constraint fk\_participa\_student references student(cod\_student),

CONSTRAINT pk\_participa PRIMARY KEY(cod\_angajat, cod\_disciplina, cod\_student)

);

CREATE TABLE studiaza(

cod\_student number(5) constraint fk\_studiaza\_student references student(cod\_student),

cod\_facultate number(5) constraint fk\_studiaza\_facultate references facultate(cod\_facultate),

CONSTRAINT pk\_studiaza PRIMARY KEY(cod\_student, cod\_facultate)

);

CREATE TABLE programa(

cod\_disciplina number(5) constraint fk\_programa\_disciplina references disciplina(cod\_disciplina),

cod\_specializare number(5) constraint fk\_programa\_specializare references specializare(cod\_specializare),

CONSTRAINT pk\_programa PRIMARY KEY(cod\_disciplina, cod\_specializare)

);

**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

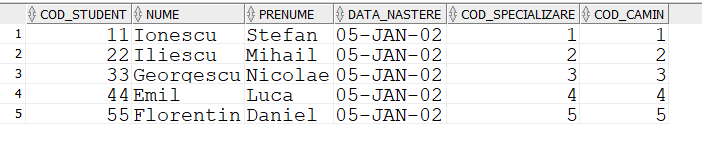
Cerinta: Pentru fiecare student, sa se afiseze numele concatenat cu prenumele, numarul de ani necesari pentru a completa specializarea la care sunt inscrisi, precum si denumirea facultatii care contine aceasta specializare si numarul de camere al caminului la care studentul este cazat.

select nume || ' ' || prenume "Nume Student", s.ani "Numarul de ani", f.denumire "Denumirea Facultatii", c.nr\_camere "Numarul de Camere"

from student e join specializare s ON(e.cod\_specializare = s.cod\_specializare)

join facultate f ON(s.cod\_facultate = f.cod\_facultate)

join camin c ON(e.cod\_camin = c.cod\_camin);



* filtrare la nivel de linii

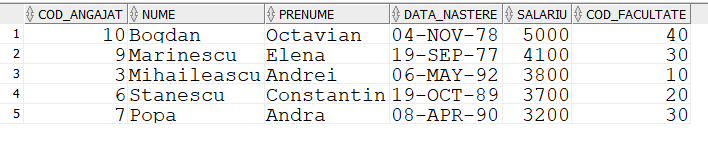
Cerinta: Sa se afiseze toti angajatii al caror salariu depaseste strict 2500 lei. Se vor ordona angajatii in ordine descrescatoare dupa salariu.

select \*

from angajat

where salariu > 2500

order by salariu desc;



* 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)

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

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

INSERT INTO angajat

VALUES(1, 'Popescu', 'Ion', '17-JAN-87', 1500, 10);

INSERT INTO angajat

VALUES(2, 'Stefanescu', 'Teodora', '25-DEC-86', 2500, 10);

INSERT INTO angajat

VALUES(3, 'Mihaileascu', 'Andrei', '06-MAY-92', 3800, 10);

INSERT INTO angajat

VALUES(4, 'Popovoci', 'Alexandra', '21-AUG-82', 1800, 20);

INSERT INTO angajat

VALUES(5, 'Ionescu', 'Mihai', '12-SEP-96', 2300, 20);

INSERT INTO angajat

VALUES(6, 'Stanescu', 'Constantin', '19-OCT-89', 3700, 20);

INSERT INTO angajat

VALUES(7, 'Popa', 'Andra', '08-APR-90', 3200, 30);

INSERT INTO angajat

VALUES(8, 'Mihai', 'George', '27-JUL-82', 1900, 30);

INSERT INTO angajat

VALUES(9, 'Marinescu', 'Elena', '19-SEP-77', 4100, 30);

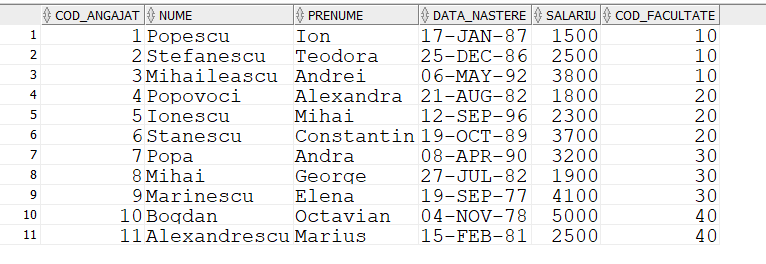
INSERT INTO angajat

VALUES(10, 'Bogdan', 'Octavian', '04-NOV-78', 5000, 40);

INSERT INTO angajat

VALUES(11, 'Alexandrescu', 'Marius', '15-FEB-81', 2500, 40);

commit;



INSERT INTO secretara

VALUES(1, 'Powerpoint', 'Prezentari');

INSERT INTO secretara

VALUES(3, 'Word', 'Birocratie');

INSERT INTO secretara

VALUES(5, 'Excel', 'Organizare');

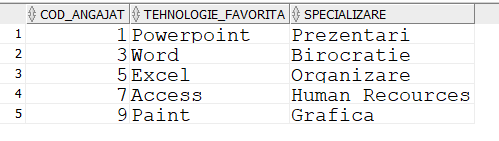
INSERT INTO secretara

VALUES(7, 'Access', 'Human Recources');

INSERT INTO secretara

VALUES(9, 'Paint', 'Grafica');

commit;



INSERT INTO profesor

VALUES(2, 9.25, 'Cursant');

INSERT INTO profesor

VALUES(4, 5.23, 'Seminarist');

INSERT INTO profesor

VALUES(6, 7.5, 'Laborant');

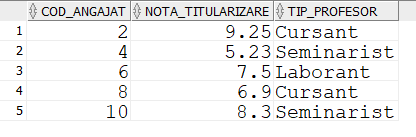
INSERT INTO profesor

VALUES(8, 6.9, 'Cursant');

INSERT INTO profesor

VALUES(10, 8.3, 'Seminarist');

commit;



INSERT INTO facultate

VALUES(10, 'Facultatea de Matematica si Informatica', 1);

INSERT INTO facultate

VALUES(20, 'Facultatea de Automatica si Calculatoare', 2);

INSERT INTO facultate

VALUES(30, 'Facultatea de Agronomie', 3);

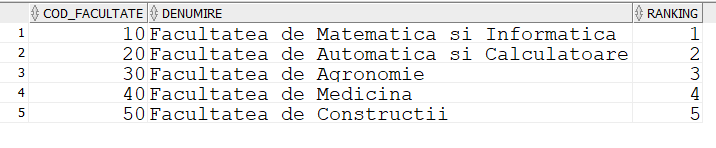
INSERT INTO facultate

VALUES(40, 'Facultatea de Medicina', 4);

INSERT INTO facultate

VALUES(50, 'Facultatea de Constructii', 5);

commit;



INSERT INTO disciplina

VALUES(11, 'Matematica', 5);

INSERT INTO disciplina

VALUES(12, 'Chimie', 6);

INSERT INTO disciplina

VALUES(13, 'Geografie', 2);

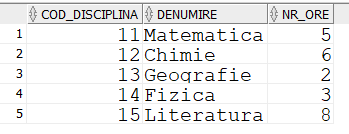
INSERT INTO disciplina

VALUES(14, 'Fizica', 3);

INSERT INTO disciplina

VALUES(15, 'Literatura', '8');

commit;



INSERT INTO examinare

VALUES(1, 11, 'Examen', 10);

INSERT INTO examinare

VALUES(2, 12, 'Proiect', 9.5);

INSERT INTO examinare

VALUES(3, 13, 'Ascultare Orala', 5.77);

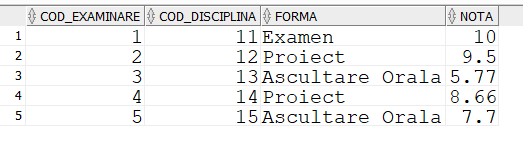
INSERT INTO examinare

VALUES(4, 14, 'Proiect', 8.66);

INSERT INTO examinare

VALUES(5, 15, 'Ascultare Orala', 7.7);

commit;



INSERT INTO specializare

VALUES(1, 'Informatica', 3, 10);

INSERT INTO specializare

VALUES(2, 'Matematica', 4, 20);

INSERT INTO specializare

VALUES(3, 'Finante', 5, 30);

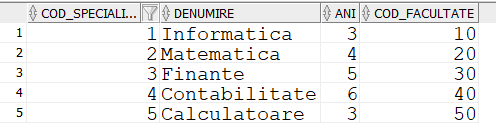
INSERT INTO specializare

VALUES(4, 'Contabilitate', 6, 40);

INSERT INTO specializare

VALUES(5, 'Calculatoare', 3, 50);

commit;



INSERT INTO camin

VALUES(1, 'Grozavesti', 50);

INSERT INTO camin

VALUES(2, 'Politehnica', 600);

INSERT INTO camin

VALUES(3, 'Magic Dorm', 120);

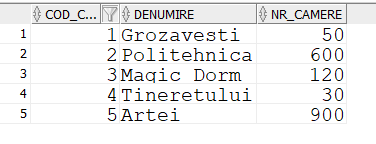
INSERT INTO camin

VALUES(4, 'Tineretului', 30);

INSERT INTO camin

VALUES(5, 'Artei', 900);

commit;



INSERT INTO student

VALUES(11, 'Ionescu', 'Stefan', '05-JAN-02', 1, 1);

INSERT INTO student

VALUES(22, 'Iliescu', 'Mihail', '05-JAN-02', 2, 2);

INSERT INTO student

VALUES(33, 'Georgescu', 'Nicolae', '05-JAN-02', 3, 3);

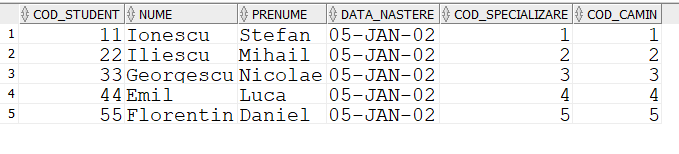
INSERT INTO student

VALUES(44, 'Emil', 'Luca', '05-JAN-02', 4, 4);

INSERT INTO student

VALUES(55, 'Florentin', 'Daniel', '05-JAN-02', 5, 5);

commit;



INSERT INTO tara

VALUES(1, 'Romania', 'Europa');

INSERT INTO tara

VALUES(2, 'Argentina', 'America de Sud');

INSERT INTO tara

VALUES(3, 'SUA', 'America de Nord');

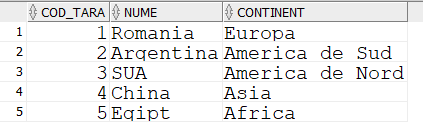
INSERT INTO tara

VALUES(4, 'China', 'Asia');

INSERT INTO tara

VALUES(5, 'Egipt', 'Africa');

commit;



INSERT INTO locatie

VALUES(1, 'Mihai Eminescu', 'Bucuresti', 1);

INSERT INTO locatie

VALUES(2, 'Mihail Kogalniceanu', 'Craiova', 2);

INSERT INTO locatie

VALUES(3, 'George Enescu', 'Timisoara', 3);

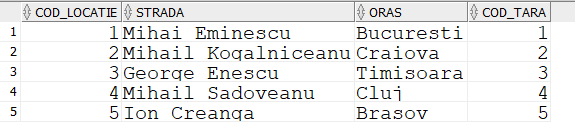
INSERT INTO locatie

VALUES(4, 'Mihail Sadoveanu', 'Cluj', 4);

INSERT INTO locatie

VALUES(5, 'Ion Creanga', 'Brasov', 5);

commit;



INSERT INTO participa

VALUES(2, 11, 11);

INSERT INTO participa

VALUES(2, 11, 22);

INSERT INTO participa

VALUES(2, 11, 33);

INSERT INTO participa

VALUES(4, 12, 11);

INSERT INTO participa

VALUES(4, 12, 44);

INSERT INTO participa

VALUES(6, 13, 55);

INSERT INTO participa

VALUES(6, 13, 33);

INSERT INTO participa

VALUES(8, 14, 44);

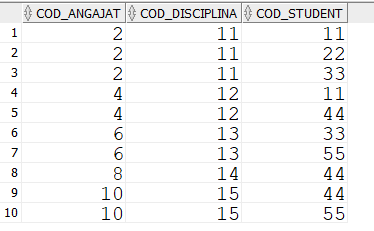
INSERT INTO participa

VALUES(10, 15, 44);

INSERT INTO participa

VALUES(10, 15, 55);

commit;



INSERT INTO studiaza

VALUES(11, 10);

INSERT INTO studiaza

VALUES(11, 20);

INSERT INTO studiaza

VALUES(22, 20);

INSERT INTO studiaza

VALUES(22, 30);

INSERT INTO studiaza

VALUES(33, 20);

INSERT INTO studiaza

VALUES(33, 30);

INSERT INTO studiaza

VALUES(33, 40);

INSERT INTO studiaza

VALUES(44, 40);

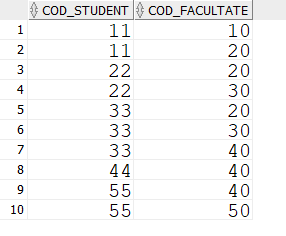
INSERT INTO studiaza

VALUES(55, 40);

INSERT INTO studiaza

VALUES(55, 50);

commit;



INSERT INTO programa

VALUES(11, 1);

INSERT INTO programa

VALUES(12, 1);

INSERT INTO programa

VALUES(12, 2);

INSERT INTO programa

VALUES(12, 3);

INSERT INTO programa

VALUES(13, 1);

INSERT INTO programa

VALUES(13, 3);

INSERT INTO programa

VALUES(14, 2);

INSERT INTO programa

VALUES(14, 3);

INSERT INTO programa

VALUES(15, 4);

INSERT INTO programa

VALUES(15, 5);

commit;

**16. Formulați în limbaj natural și implementați în SQL: o cerere ce utilizează operația outerjoin pe minimum 4 tabele și două cereri ce utilizează operația division.**

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

**18. a. Realizarea normalizării BCNF, FN4, FN5.**

**b. Aplicarea denormalizării, justificând necesitatea acesteia.**