

Proiect Sisteme de Gestiune a Bazelor de Date

Baza de Date pentru Gestiunea unei Universitati

Nume: Kayed Amar

Grupa: 243

Email: amar.kayed@s.unibuc.ro

Anul Universitar 2021 - 2022

1.

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 alcatauita 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. Există 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 diferențiază prin cel putin specializarea si profesorul care o predă.

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 stabilita 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 aparținand 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.

Prezentarea constrângerilor (restricții, reguli) impuse asupra modelului.

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

Modelul de date respectă anumite restricții de funcționare.

- 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 specializare.
- 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 aparțin aceleasi universitatii, de aici motivandu-se absenta unei entitati pentru universitatii.
- Cu toate ca toate facultatile aparțin aceleasi universitatii, acestea pot prezenta specializari/discipline asemănătoare 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.

Descrierea entităților, incluzând precizarea cheii primare.

Pentru modelul de date referitor la gestiunea unei universități, structurile ANGAJAT, SECRETARA, PROFESOR, DISCIPLINA, EXAMINARE, STUDENT, CAMIN, SPECIALIZARE, FACULTATE, LOCATIE, TARA reprezintă entități.

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

Toate entitățile prezentate sunt independente, cu excepția entității EXAMINARE, aceasta depinzând de entitatea DISCIPLINA (nu poate exista examinare dacă nu există disciplina/materie pentru acest examen).

1. ANGAJAT = persoana fizică/salariat în cadrul unei facultăți și, implicit, în cadrul unei universități. Angajatul se ocupă cu organizarea și gestionarea diferitelor componente care intra buna funcționare a unei facultăți. Această entitate reprezintă o superclasa pentru entitățile PROFESOR și SECRETARA, întrucât un angajat în cadrul unei facultăți poate fi un profesor sau o secretară, sau niciuna. Cheia primară a acestei superentități este *cod_angajat*.
2. SECRETARA = subentitate a entității ANGAJAT, aceasta reprezintă reprezintă o persoană fizică care se ocupă cu administrarea unei facultăți, cu organizarea și întreținerea activităților academice precum și a componentelor birocratice/juridice. Fiind o subentitate, aceasta mosteneste cheia primară a superclasei ANGAJAT, anume atributul *cod_angajat*.
3. PROFESOR = subentitate a entității ANGAJAT, aceasta reprezintă un angajat care se ocupă cu predarea disciplinelor din cadrul unei specializări a facultății, precum și de evaluarea studentilor înscriși la facultatea respectivă. Fiind o subentitate, aceasta mosteneste cheia primară a superclasei ANGAJAT, anume atributul *cod_angajat*.
4. DISCIPLINA = concept academic în cadrul unei facultăți, aceasta reprezintă un set de competențe menite să fie predăte studentilor, urmand ca această să asimileze cunoștiința și să fie evaluate din respectivele competențe. Cheia primară a entității este *cod_disciplina*.
5. EXAMINARE = activitate de evaluare a competențelor dobândite de student în cadrul unei discipline. Această entitate este dependenta de entitatea DISCIPLINA, care conține informații

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 predă respectiva disciplina. Cheia primara a acestei entitati este *cod_student*(initial voi am 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 institutionalala 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*.

Descrierea relațiilor, incluzând precizarea cardinalității acestora.

Vom prezenta relațiile modelului de date, dând o descriere completă a fiecărei. De fapt, denumirile acestor legături sunt sugestive, reflectând conținutul acestora și entitățile pe care le leagă. Pentru fiecare relație se va preciza cardinalitatea minimă și maximă.

- ANGAJAT ISA SECRETARA = relație 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 contine minim un angajat(altfel s-ar desfiinta facultatea). De asemenea, aceasta relatia 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 relatia 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 relatia 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

contine 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 aceeași 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 predată 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 predă disciplina la care **participa** un student. Datorita acestei relatii de tip 3, acum putem raspunde la intrebarea “la ce disciplina particip un student pentru un anumit profesor”, intrucat o relatie de tip 3 este diferita de 3 relatii de tip 2.

Descrierea atributelor, incluzând tipul de date și eventualele constrângeri, valori implicate, 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. Entitatea 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, subentitatea 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 profesoari 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. Entitatea 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 lungime 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 predante 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.

- ***cod_locatie*** = variabila de tip intreg, de lungime maxima 5, care reprezinta codul locatiei in care se afla facultatea. Acest atribut este o cheie externa care refera cheia primara din LOCATIE. 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 tarii. 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 relatia 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.

2.

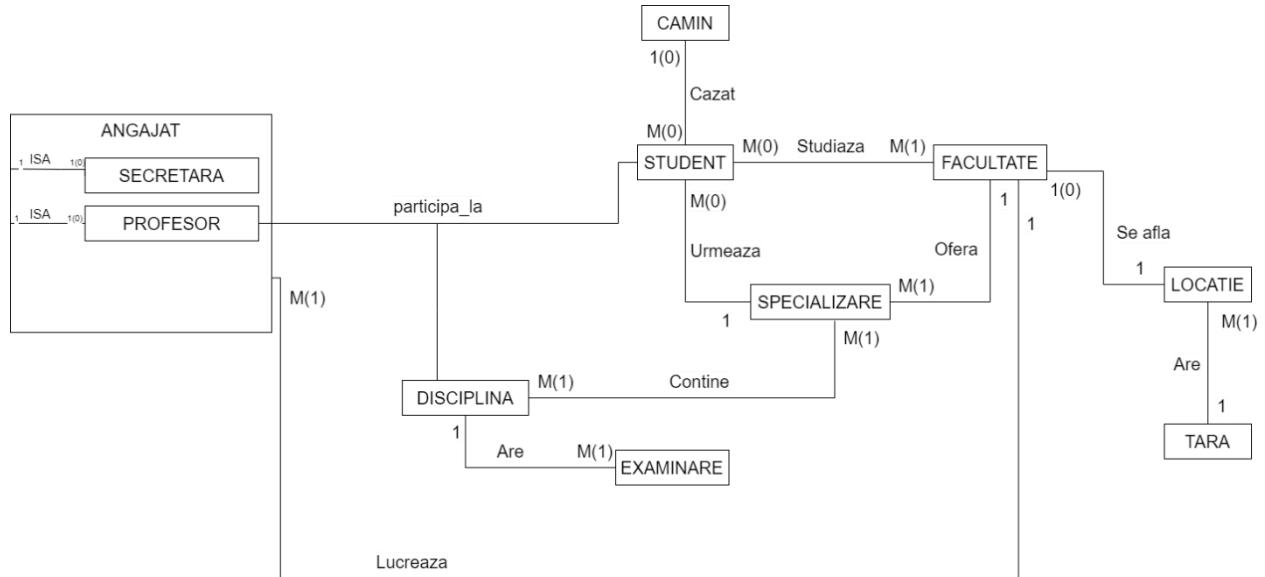
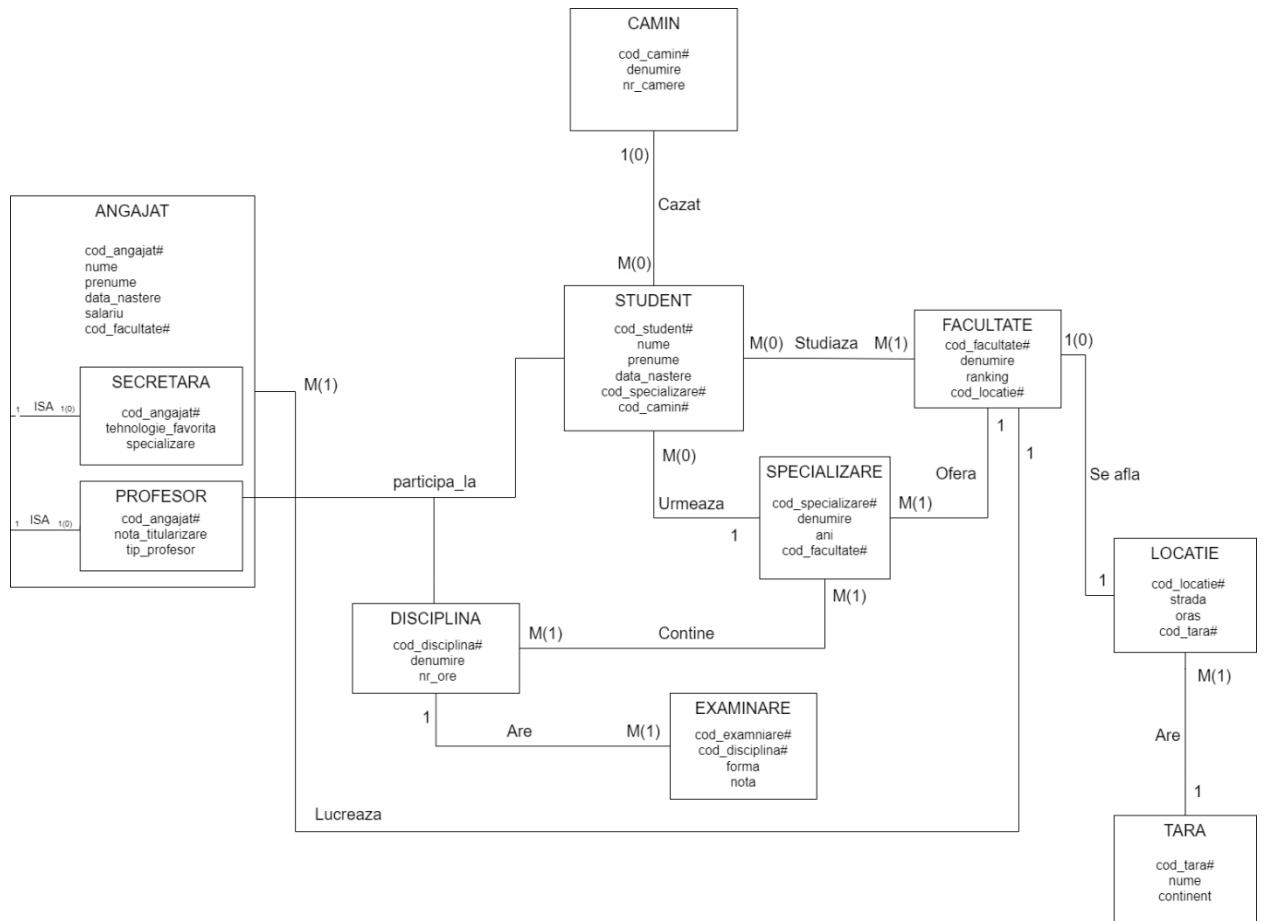


Diagrama ER cu atribute:



3.

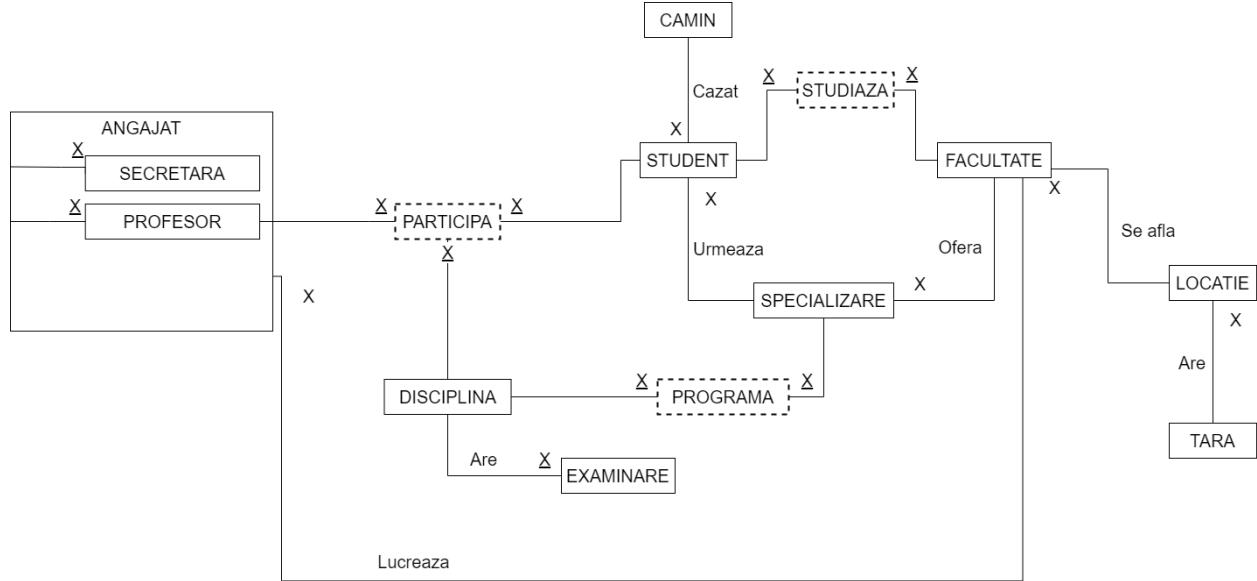
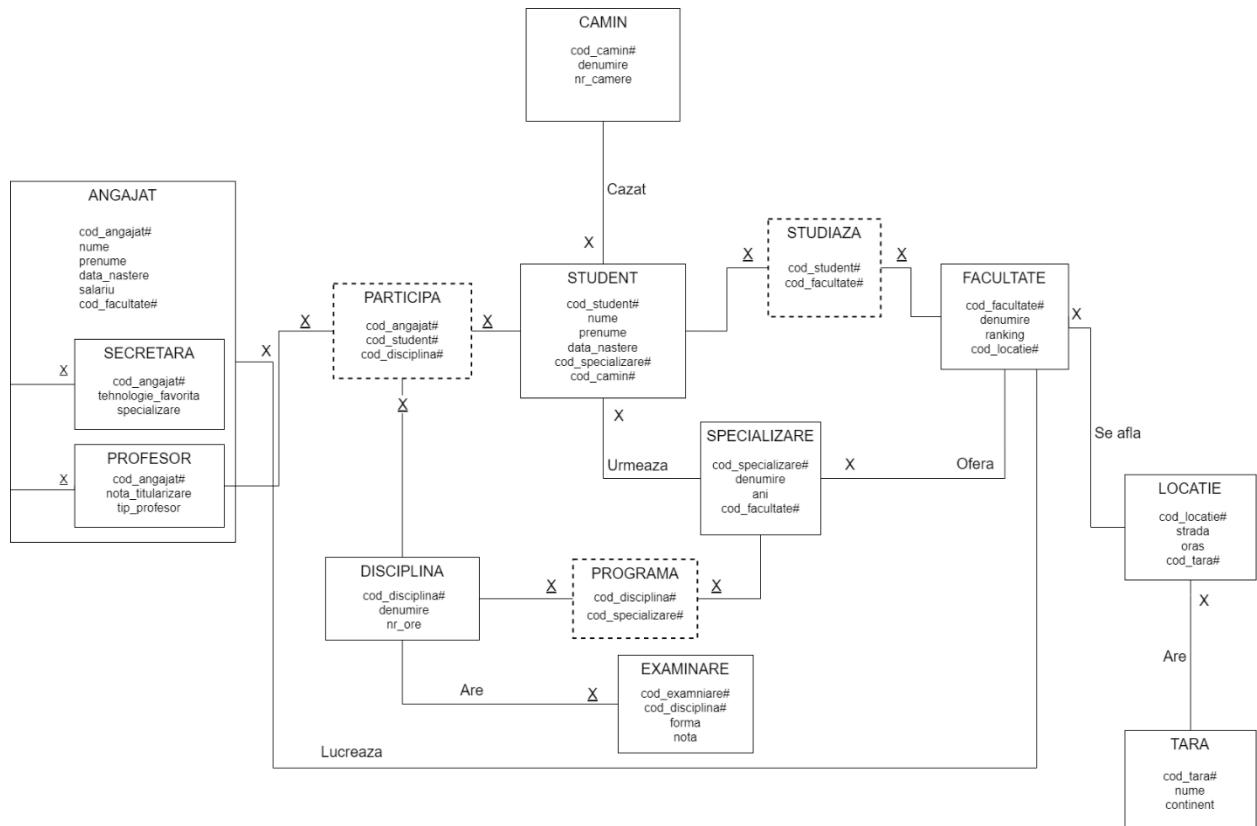


Diagrama Conceptuala cu atribute:



4.

```
CREATE TABLE tara(
    cod_tara number(5) constraint pk_tara primary key,
    nume varchar2(25),
    continent varchar2(25)
);

CREATE TABLE locatie(
    cod_locatie number(5) constraint pk_locatie primary key,
    strada varchar2(25),
    oras varchar2(25),
    cod_tara number(5),
    constraint fk_locatie_tara foreign key(cod_tara) references tara(cod_tara)
);

CREATE TABLE facultate(
    cod_facultate NUMBER(5) CONSTRAINT pk_facultate PRIMARY KEY,
    denumire VARCHAR2(50) CONSTRAINT denumire_facultate NOT NULL,
    ranking NUMBER(4),
    cod_locatie number(5) constraint fk_facultate_locatie references locatie(cod_locatie)
);

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 camin(  
    cod_camin number(5) constraint pk_camin primary key,  
    denumire varchar2(30),  
    nr_camere number(4)  
);
```

```
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 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 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 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)
```

```
);
```

```
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) CONSTRAINT cod_facultate_ang NOT NULL
```

```
/*
```

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))
```

```
*/
```

```
);
```

```
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 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)  
);
```

```
Table TARA created.

Table LOCATIE created.

Table FACULTATE created.

Table SPECIALIZARE created.

Table CAMIN created.

Table STUDENT created.

Table STUDIAZA created.

Table DISCIPLINA created.
```

5.

-- Inserari:

```
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;

	COD_TARA	NUME	CONTINENT
1	1	Romania	Europa
2	2	Argentina	America de Sud
3	3	SUA	America de Nord
4	4	China	Asia
5	5	Egipt	Africa

```
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;
```

	COD_LOCATIE	STRADA	ORAS	COD_TARA
1	1	Mihai Eminescu	Bucuresti	1
2	2	Mihail Kogalniceanu	Craiova	2
3	3	George Enescu	Timisoara	3
4	4	Mihail Sadoveanu	Cluj	4
5	5	Ion Creanga	Brasov	5

```
INSERT INTO facultate  
VALUES(10, 'Facultatea de Matematica si Informatica', 1, 2);
```

```
INSERT INTO facultate  
VALUES(20, 'Facultatea de Automatica si Calculatoare', 2, 5);
```

```
INSERT INTO facultate  
VALUES(30, 'Facultatea de Agronomie', 3, 1);
```

```
INSERT INTO facultate  
VALUES(40, 'Facultatea de Medicina', 4, 3);
```

```
INSERT INTO facultate  
VALUES(50, 'Facultatea de Constructii', 5, 4);
```

```
commit;
```

	COD_FACULTATE	DENUMIRE	RANKING	COD_LOCATIE
1	10	Facultatea de Matematica si Informatica	1	2
2	20	Facultatea de Automatica si Calculatoare	2	5
3	30	Facultatea de Agronomie	3	1
4	40	Facultatea de Medicina	4	3
5	50	Facultatea de Constructii	5	4

```
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;
```

	COD_SPECIALIZARE	DENUMIRE	ANI	COD_FACULTATE
1	1	Informatica	3	10
2	2	Matematica	4	20
3	3	Finante	5	30
4	4	Contabilitate	6	40
5	5	Calculatoare	3	50

```
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;
```

	COD_CAMIN	DENUMIRE	NR_CAMERE
1	1	Grozavesti	50
2	2	Politehnica	600
3	3	Magic Dorm	120
4	4	Tineretului	30
5	5	Artei	900

```
INSERT INTO student  
VALUES(11, 'Ionescu', 'Stefan', '05-JAN-02', 1, 1);
```

```
INSERT INTO student  
VALUES(22, 'Iliescu', 'Mihail', '12-DEC-89', 2, 2);
```

```
INSERT INTO student  
VALUES(33, 'Georgescu', 'Nicolae', '23-MAY-95', 3, 3);
```

```
INSERT INTO student  
VALUES(44, 'Emil', 'Luca', '28-APR-00', 4, 4);
```

```
INSERT INTO student  
VALUES(55, 'Florentin', 'Daniel', '19-SEP-92', 5, 5);
```

```
commit;
```

	COD_STUDENT	NUME	PRENUME	DATA_NASTERE	COD_SPECIALIZARE	COD_CAMIN
1	11	Ionescu	Stefan	05-JAN-02	1	1
2	22	Iliescu	Mihail	12-DEC-89	2	2
3	33	Georgescu	Nicolae	23-MAY-95	3	3
4	44	Emil	Luca	28-APR-00	4	4
5	55	Florentin	Daniel	19-SEP-92	5	5

```
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;
```

	COD_STUDENT	COD_FACULTATE
1	11	10
2	11	20
3	22	20
4	22	30
5	33	20
6	33	30
7	33	40
8	44	40
9	55	40
10	55	50

```
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;
```

	COD_DISCIPLINA	DENUMIRE	NR_ORE
1	11	Matematica	5
2	12	Chimie	6
3	13	Geografie	2
4	14	Fizica	3
5	15	Literatura	8

```
INSERT INTO examinare  
VALUES(1, 11, 'Examen', 10);
```

```
INSERT INTO examinare  
VALUES(2, 12, 'Proiect', 9.5);
```

```
INSERT INTO examinare  
VALUES(3, 13, 'A scultare Orala', 5.77);
```

```
INSERT INTO examinare  
VALUES(4, 14, 'Proiect', 8.66);
```

```
INSERT INTO examinare  
VALUES(5, 15, 'A scultare Orala', 7.7);
```

```
commit;
```

	COD_EXAMINARE	COD_DISCIPLINA	FORMA	NOTA
1	1	11	Examen	10
2	2	12	Proiect	9.5
3	3	13	A scultare Orala	5.77
4	4	14	Proiect	8.66
5	5	15	A scultare Orala	7.7

```
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;
```

	COD_DISCIPLINA	COD_SPECIALIZARE
1	11	1
2	12	1
3	12	2
4	12	3
5	13	1
6	13	3
7	14	2
8	14	3
9	15	4
10	15	5

```
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, 'Popovici', '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;
```

	COD_ANGAJAT	NUME	PRENUME	DATA_NASTERE	SALARIU	COD_FACULTATE
1	1	Popescu	Ion	17-JAN-87	1500	10
2	2	Stefanescu	Teodora	25-DEC-86	2500	10
3	3	Mihaileascu	Andrei	06-MAY-92	3800	10
4	4	Popovoci	Alexandra	21-AUG-82	1800	20
5	5	Ionescu	Mihai	12-SEP-96	2300	20
6	6	Stanescu	Constantin	19-OCT-89	3700	20
7	7	Popa	Andra	08-APR-90	3200	30
8	8	Mihai	George	27-JUL-82	1900	30
9	9	Marinescu	Elena	19-SEP-77	4100	30
10	10	Bogdan	Octavian	04-NOV-78	5000	40
11	11	Alexandrescu	Marius	15-FEB-81	2500	40

```
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;
```

	COD_ANGAJAT	TEHNOLOGIE_FAVORITA	SPECIALIZARE
1	1	Powerpoint	Prezentari
2	3	Word	Birocratie
3	5	Excel	Organizare
4	7	Access	Human Recources
5	9	Paint	Grafica

```
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;
```

COD_ANGAJAT	NOTA_TITULARIZARE	TIP_PROFESOR
1	2	9.25 Cursant
2	4	5.23 Seminarist
3	6	7.5 Laborant
4	8	6.9 Cursant
5	10	8.3 Seminarist

```
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;
```

	COD_ANGAJAT	COD_DISCIPLINA	COD_STUDENT
1	2	11	11
2	2	11	22
3	2	11	33
4	4	12	11
5	4	12	44
6	6	13	33
7	6	13	55
8	8	14	44
9	10	15	44
10	10	15	55

6.

/*

pentru un id dat ca parametru si incepand cu urmatoarele linii de dupa acest id pana la sfarsitul tabelului

(incepand cu acest id)

pentru fiecare student al carui id este egal cu un numar fibonacci +- o marja de eroare de maxim 1 numar

sa se afiseze numele profesorilor la care acesta invata precum si salariul acestora

afisarea se va face sub forma: nume prenume salariu

pentru fiecare profesor in parte

De asemenea, sa se afiseze lista tuturor numerelor fibonacci mai mici sau egale +- 1 decat id-ul maxim al categoriei

de studenti aflate pe liniile urmatoare din tabel fata de id-ul dat ca parametru

Daca nu exista studenti cu id-uri egale cu numere fibonacci, se va arunca o exceptie "NO_FIBONACCI_FOUND"

daca mai mult de jumata + 1 dintre toti studentii eligibili(care se afla sub id-ul dat ca parametru)

au id-uri egale cu numere fibonacci +- marja de eroare, atunci se va arunca exceptia "TOO_MANY_FIBONACCI"

Spunem ca un numar este egal +-1 cu un numar fibonacci x daca acesta este cuprins in intervalul [x-1, x+1].

Indiferent daca se va arunca o exceptie no_fibonacci_found sau too_many_fibonacci id-ul dat, id-ul maxim, precum si numerele fibonacci mai mici ca id-ul maxim vor fi afisate.

In cazul exceptiei no_data_found, se va trata direct exceptia, intrucat nu se poate face nimic fara un id valid.

Rezolvare:

Verificam daca id-ul dat ca parametru este valid, daca nu este valid ne oprim la acest pas

Calculam id-ul maxim luand in considerare doar id-urile care vin imediat dupa id-ul dat ca parametru

Folosim un tablou indexat pentru numerele fibonacci, trebuie sa gasim id-ul maxim pentru ultimul nr fibonacci

Folosim un record cu numele si salariul unui profesor

Folosim un varray de tipul record anterior mentionat pentru a retine detaliiile tuturor profesorilor la care invata un student

Folosim un tablou imbricat de varray-uri pentru a contrui o matrice in care fiecare linie i reprezinta lista profesorilor studentului i

Pentru majoritatea operatiilor vom folosi un cursor pentru a parurge tabelul student

*/

-- Cod Exercitiul 6:

```
create or replace procedure exercitiul_6(
```

```
    id      student.cod_student%type      -- codul studentului dat ca parametru
)
is
```

```
    cursor c is select rownum, cod_student, nume, prenume from student;    -- cursor pentru a
parcurge liniile din tabelul student
```

```
    nr_linie_id      number;           -- numarul liniei parcurse de cursor
    id_student       student.cod_student%type;      -- id-ul studentului parcurs de
cursor
    nume            student.nume%type;           -- numele studentului parcurs de
cursor
    prenume          student.prenume%type;         -- prenumele studentului parcurs
de cursor
```

```
    id_maxim        student.cod_student%type;      -- id-ul de valoare maxima a studentilor aflati
pe liniile urmatoare id-ului de student dat ca parametru
```

```
    type tab_ind_fibonacci is table of number index by pls_integer;      -- tablou indexat ce
memoreaza numerele fibonacci mai mici +1 decat id_maxim
```

```
    ind      tab_ind_fibonacci;           -- tabloul indexat in care o sa memoram numerele
fibonacci calculate
```

```
    contor    number;                  -- contor folosit pentru a parurge tabloul indexat
anterior declarat
```

```

nr_studenti    number;           -- numarul total de studenti
sem          boolean;           -- semafor folosit pentru eficientizarea timpului de
executie

no_fibonacci_found exception;   -- exceptie specifica exercitiului
too_many_fibonacci exception;  -- exceptie specifica exercitiului

type detalii_profesor is record(      -- tip inregistrare(record) pentru memorarea detaliilor
despre un profesor
    nume    angajat.nume%type,
    prenume  angajat.prenume%type,
    salariu angajat.salariu%type
);

type v_detalii_profesor is varray(25) of detalii_profesor;  -- varray de 25 de detalii_profesor

type tab_imb_nume_profesori is table of v_detalii_profesor; -- tablou imbricat al carui
elemente sunt varray-uri de detalii_profesor(matrice)

imb tab_imb_nume_profesori := tab_imb_nume_profesori();       -- imb este practic o
matrice de detalii_profesor

begin
    -- verificam daca id-ul dat ca parametru exista de fapt in baza de date
    select cod_student into id_student from student where cod_student = id;
    -- generam NO_DATA_FOUND daca nu gasim id-ul dat ca parametru
    -- Daca nu s-a generat eroarea NO_DATA_FOUND inseamna ca id-ul exista si il avem memorat
    in variabila auxiliara id_student

```

```

open c; -- deschidem cursorul pentru a incepe parcurgerea tabelului student

loop -- cat timp nu am parcurs tot tabelul

    fetch c into nr_linie_id, id_student, nume, prenume; -- plasam valorile liniei curente in
    variabilele asociate

        exit when c%notfound; -- oprim bucla atunci cand am trecut de ultima
        linie a tabelului

        if id_student = id then -- daca id-ul curent este egal cu id-ul dat ca
        parametru

            exit; -- inseamna ca putem sa iesim din bucla parcurgerii
            cursorului

        end if;

    end loop;

-- am gasit linia unde se afla student-ul cu id-ul dat
-- trebuie sa calculam id-ul maxim dintre cei ramasi

id_maxim := id_student; -- initial, id-ul maxim va fi fix id-ul dat ca parametru

loop -- incepand cu linia imediat urmatoare liniei in care se afla id-ul
dat ca parametru

    exit when c%notfound; -- verificam intai daca id-ul dat ca parametru se afla
    deja pe ultima linie a tabelului student

    fetch c into nr_linie_id, id_student, nume, prenume; -- Daca nu era pe ultima linie,
    atunci incepem cautarea id-ului maxim

    if id_student > id_maxim then -- Daca id-ul curent este mai mare decat id-ul maxim
    calculat pana in acest moment

        id_maxim := id_student; -- Atunci noul id_maxim devine id-ul curent

```

```

end if;

end loop;

dbms_output.put_line('ID-ul dat: ' || id);      -- Printam id-ul dat ca parametru
dbms_output.put_line('ID-ul maxim: ' || id_maxim); -- Printam id-ul maxim incepand cu
linia urmatoare id-ului curent
close c;

dbms_output.new_line;      -- Printam o noua linie

-- Acum ca avem id-ul maxim calculat, putem incepe calcularea numerelor fibonacci mai mici
+-1 decat id_maxim

if 1 <= id_maxim + 1 then    -- Daca primul termen fibonacci(1) este mai mic sau egal decat
id_maxim + 1 atunci putem incepe calcularea
    ind(1) := 1;            -- Primul termen fibonacci este 1
    ind(2) := 1;            -- Al doilea termen fibonacci este 2
    contor := 3;            -- Calculul incepe cu al treilea termen fibonacci, deci contor este 3
else        -- Daca in schimb id_maxim + 1 este mai mic decat primul termen fibonacci,
atunci nu avem ce termeni fibonacci sa calculam(numerele fibonacci sunt strict pozitive)
    raise no_fibonacci_found;    -- Prin urmare, aruncam exceptia no_fibonacci_found,
intrucat nu avem numere fibonacci
end if;

-- Daca nu s-a aruncat exceptia, atunci putem continua calcularea numerelor fibonacci
while ind(contor-1) + 1 <= id_maxim loop        -- Cat timp termenul anterior + 1 este
mai mic sau egal decat id_maxim calculam urmatorul termen fibonacci

```

```

if ind(contor - 1) + ind(contor - 2) > id_maxim then -- Daca urmatorul termen fibonacci va
fi mai mare strict decat id_maxim, nu il mai calculam

exit;                                -- Ci in schimb iesim din bucla

end if;

ind(contor) := ind(contor-1) + ind(contor-2); -- fib(i) = fib(i-1) + fib(i-2);

contor := contor + 1;                  -- i++;

end loop;

```

dbms_output.put_line('Numerele fibonacci mai mici ca ' || id_maxim || ':'); -- Incepem afisarea termenilor fibonacci calculati

```

for i in ind.first..ind.last loop      -- Parcurgem tabloul indexat folosind o bucla for

    dbms_output.put(ind(i) || ' ');     -- Printam toti termenii fibonacci intr-o singura linie
separand prin spatiu

end loop;

dbms_output.new_line;                 -- Afisam 3 linii noi pentru a separa continutul afisat
dbms_output.new_line;
dbms_output.new_line;

```

-- Determinam numarul de studenti care se afla dupa id-ul dat, precum si numarul de studenti al caror id este fibonacci

```

nr_studenti := 0;                    -- Initial consideram ca numarul de studenti aflati sub id-ul dat este 0

contor := 0;                         -- In contor vom memora numarul de id-uri fibonacci, contor este
initializat cu 0

open c;                             -- Deschidem cursorul din nou, acesta se afla inainte de prima linie din
tabel

loop                                -- Cautam sa ajungem la linia unde se afla id-ul dat ca parametru

    fetch c into nr_linie_id, id_student, nume, prenume;

```

```

    exit when id_student = id; -- Odata ce am ajuns la linia cu id-ul dat ca parametru oprim
bucla

    end loop;

    loop          -- Incepem sa numaram studentii aflati sub id-ul dat

        nr_studenti := nr_studenti + 1; -- Luam in considerare si id-ul dat ca fiind unul elegibil

        -- Testam daca id-ul curent este fibonacci

        sem := true;           -- Consideram sem = true atunci cand id-ul nu este fibonacci

        for i in ind.first..ind.last loop      -- Parcurgem tabloul de numere fibonacci

            if id_student >= ind(i) - 1 and id_student <= ind(i) + 1 then    -- Daca id-ul curent este
egal +-1 fata de termenul fibonacci curent

                contor := contor + 1;   -- Incrementam contor-ul pentru a creste numarul de id-uri
fibonacci

                sem := false;         -- sem devine false, intrucat stim ca id-ul este fibonacci

                end if;

            if sem = false then      -- Daca am determinat ca id-ul este fibonacci, nu mai este
nevoie sa comparam cu restul numerelor fibonacci din tablou

                exit;             -- Prin urmare, putem iesi din bucla for

                end if;

            end loop;

        fetch c into nr_linie_id, id_student, nume, prenume; -- Trecem la urmatorul id
        exit when c%notfound;           -- Repetam procesul pentru fiecare id in
parte

        end loop;

```

```
close c; -- Inchidem cursorul odata ce am terminat
```

```
/*
```

Pana acum avem urmatoarele:

```
contor = numarul de studenti cu id fibonacci
```

```
nr_studenti = numarul tuturor studentilor
```

```
*/
```

```
-- dbms_output.put_line(contor);
```

```
-- dbms_output.put_line(nr_studenti);
```

```
if contor = 0 then          -- daca contor = 0 inseaman ca nu avem id-uri fibonacci si  
aruncam exceptia no_fibonacci_found
```

```
    raise no_fibonacci_found;
```

```
elsif contor > 0.5 * nr_studenti then    -- daca avem mai mult de jumata + 1 dintre toti  
studentii eligibil cu id-uri fibonacci, intram pe exceptia TOO_MANY_FIBONACCI
```

```
    raise too_many_fibonacci;
```

```
end if;
```

```
-- daca nu am intrat pe exceptia TOO_MANY_FIBONACCI, inseamna putem sa
```

```
-- afisam toti studentii care se incadreaza in conditia id-ului:
```

```
-- Practic nu mai avem nevoie de numarul de id-uri fibonacci, deci putem sa reutilizam  
variabila contor
```

```
open c; -- Deschidem cursorul pentru parcurgerea tabelului student
```

```
loop
```

```
    fetch c into nr_linie_id, id_student, nume, prenume;
```

```
    exit when id_student = id;      -- Parcurgem toate liniile pana la linia cu id-ul dat ca  
parametru
```

```

end loop;

-- Incepem sa construim matricea de detalii_profesor din imb
contor := 1; -- Contor ia primul indice al tabloului imb

loop
    -- Pentru fiecare id de dupa id-ul parametru, verificam daca este fibonacci si afisam
datele
    for i in ind.first..ind.last loop
        if id_student >= ind(i) - 1 and id_student <= ind(i) + 1 then -- Daca id-ul este id
fibonacci afisam informatiile
            dbms_output.put_line('Nr linie: ' || nr_linie_id);
            dbms_output.put_line('ID: ' || id_student);
            dbms_output.put_line('Nume Student: ' || nume || ' ' || prenume);
            dbms_output.put('Profesori: ');
    end loop;
    imb.extend; -- necesar tablourilor imbriicate si varray-urilor

    select a.nume, a.prenume, a.salariu      -- Selectam toti profesorii studentului cu
id-ul curent
        bulk collect into imb(contor)          -- Stocam toate campurile selectate in
imb(contor), care este de tipul varray(25) de detalii_profesor
        from participa p join student s on (s.cod_student = p.cod_student)
            join profesor pf on (p.cod_angajat = pf.cod_angajat)
            join angajat a on (p.cod_angajat = a.cod_angajat)
        where s.cod_student = id_student
        order by s.nume;

```

```
for i in imb(contor).first..imb(contor).last loop      -- Parcurgem lista tuturor profesorilor unui student

    dbms_output.put(imb(contor)(i).nume || '' || imb(contor)(i).prenume || '' || imb(contor)(i).salariu); -- Afisam numele, prenumele si salariul profesorului curent

    if i < imb(contor).last then      -- Daca nu am ajuns la ultimul profesor

        dbms_output.put(', ');      -- Afisam virgula ca separator

    else

        dbms_output.put('.');      -- Daca am ajuns la ultimul profesor, punem simbolul punct.

    end if;

end loop;
```

```
contor := contor + 1;      -- Incrementam contor-ul pentru a trece la urmatorul element al lui imb
```

```
dbms_output.new_line;      -- Punem new_line pentru a se afisa toate dbms_output.put()-urile executate in for

dbms_output.new_line;      -- Punem inca un new_line pentru a pastra afisarea mai spatiata
```

```
end if;
```

```
end loop;
```

```
fetch c into nr_linie_id, id_student, nume, prenume;      -- Trecem la urmatorul student
```

```
    exit when c%notfound;          -- Incheiem bucla atunci cand am terminat
studentii
```

```
end loop;
```

```
close c;
```

```
ind.delete;    -- stergem tabloul indexat
for i in imb.first..imb.last loop
    imb(i).delete;    -- Stergem fiecare varray existent in imb
end loop;
imb.delete;      -- Stergem tabloul imbricat
```

```
exception
```

```
when no_data_found then dbms_output.put_line('ID-ul dat nu exista in baza de date');    --
Atunci cand avem no_data_found inseamna ca nu exista student cu id-ul dat ca parametru
```

```
when no_fibonacci_found then dbms_output.put_line('Niciun student nu are id fibonacci');
-- Atunci cand nu exista niciun id fibonacci sau daca id_maxim este mai mic decat primul termen
fibonacci
```

```
when too_many_fibonacci then dbms_output.put_line('Mai mult de jumatate + 1 dintre
studentii eligibili au id-uri fibonacci'); -- Atunci cand mai mult de jumatate + 1 dintre id-uri sunt
fibonacci primim aceasta eroare
```

```
-- nu intalnim erori de tipul too_many_rows, intrucat toate select-urile pe care le
-- efectuam se fac impreuna cu clauza where cod_student = id;
-- din moment ce prin definitie cod_student este cheie primara in tabela student
-- o sa avem cel mult un rezultat in urma select-ului.
```

```
-- when too_many_rows then dbms_output.put_line('Too Many Rows');
```

```
when others then dbms_output.put_line('Alt tip de eroare!');      -- Tratam orice alt tip de eroare
```

```
end exercitiul_6;
```

```
/
```

The screenshot shows the Oracle SQL Developer interface with the following details:

- Title Bar:** Oracle SQL Developer : C:\Users\amer2\OneDrive\Desktop\Aplicati\Facultate\Anul 2\Sisteme de Gestire a Bazelor de Date\Project\Project SGBD\ex6.sql
- Toolbar:** File Edit View Navigate Run Source Team Tools Window Help
- Left Sidebar:** Databases, Oracle Co., Oracle Database
- Central Area:** Worksheet tab (Query Builder). The code is as follows:

```
211    imb.delete;           -- stergem tabloul indexat
212    for i in imb.first..imb.last loop
213        imb(i).delete;      -- Stergem fiecare varray existent in imb
214    end loop;
215    imb.delete;           -- Stergem tabloul imbricat
216
217
218 exception
219     when no_data_found then dbms_output.put_line('ID-ul dat nu exista in baza de date');      -- Atunci cand avem no_data_found
220     when no_fibonacci_found then dbms_output.put_line('Niciun student nu are id fibonacci');      -- Atunci cand nu exista niciun
221     when too_many_fibonacci then dbms_output.put_line('Mai mult de jumata + 1 dintre studentii eligibili au id-uri fibonacci')
222     -- nu intalnim erori de tipul too_many_rows, intrucat toate select-urile pe care le
223     -- efectuam se fac impreuna cu clauza where cod_student = id;
224     -- din moment ce prin definitie cod_student este cheie primara in tabela student
225     -- o sa avem cel mult un rezultat in urma select-ului.
226
227     -- when too_many_rows then dbms_output.put_line('Too Many Rows');
228
229     when others then dbms_output.put_line('Alt tip de eroare!');      -- Tratam orice alt tip de eroare
230
231
232 end exercitiul_6;
233
```

The code handles various database errors and specific conditions related to student IDs and Fibonacci numbers.

Script Output: Task completed in 0.071 seconds

Compiler - Log: Procedure EXERCITIUL_6 compiled

Bottom Status Bar: Line 233 Column 2 | Insert | Windows CE | 2:07 AM | ENG | 26-Dec-21

-- Rezultate/Explicatii:

```
/*
```

Pentru a vedea mai bine cum se executa programul, vom insera inca o valoare oarecare al carui id nu este numar fibonacci

```
*/
```

```
insert into student
```

```
values(66, 'c', 'c', sysdate, 1, 1);
```

```
select * from student;
```

	COD_STUDENT	NUME	PRENUME	DATA_NASTERE	COD_SPECIALIZARE	COD_CAMIN
1	11	Ionescu	Stefan	05-JAN-02	1	1
2	22	Iliescu	Mihail	12-DEC-89	2	2
3	33	Georgescu	Nicolae	23-MAY-95	3	3
4	44	Emil	Luca	28-APR-00	4	4
5	55	Florentin	Daniel	19-SEP-92	5	5
6	66	c	c	26-DEC-21	1	1

```
/*
```

De asemenea, vom selecta si informatiile despre toti profesorii pentru a le avea la dispozitie.

```
*/
```

```
select p.cod_angajat, nume, prenume, salariu
```

```
from angajat a join profesor p on (a.cod_angajat = p.cod_angajat);
```

	COD_ANGAJAT	NUME	PRENUME	SALARIU
1	2	Stefanescu	Teodora	2500
2	4	Popovici	Alexandra	1800
3	6	Stanescu	Constantin	3700
4	8	Mihai	George	1900
5	10	Boqdan	Octavian	5000

```
execute exercitiul_6(11);
```

```
-- rezultat bun
```

```
235 insert into student  
236 values(66, 'c', 'c', sysdate, 1, 1);  
237  
238 select * from student;  
239  
240  
241 execute exercitiul_6(11);  
242 -- rezultat bun  
243 execute exercitiul_6(33);  
244 -- rezultat bun  
245 execute exercitiul_6(44);  
246 -- rezultat bun
```

COD_STUDENT	NUME	PRENUME	DATA_NASTERE	COD_SPECALIZARE	COD_CAMPN
1	11Ionescu	Stefan	05-JAN-02	1	1
2	22Iliescu	Mihail	12-DEC-89	2	2
3	33Georgescu	Nicolae	23-MAY-95	3	3
4	44Emil	Luca	28-APR-00	4	4
5	55Florentin	Daniel	19-SEP-92	5	5
6	66c	c	26-DEC-21	1	1

```
ID-ul dat: 11  
ID-ul maxim: 66  
Numerele fibonacci mai mici ca 66:  
1 1 2 3 5 8 13 21 34 55  
  
Nr linie: 2  
ID: 22  
Nume Student: Iliescu Mihail  
Profesori: Stefanescu Teodora 2500.  
  
Nr linie: 3  
ID: 33  
Nume Student: Georgescu Nicolae  
Profesori: Stanescu Constantin 3700, Stefanescu Teodora 2500.  
  
Nr linie: 5  
ID: 55  
Nume Student: Florentin Daniel  
Profesori: Bogdan Octavian 5000, Stanescu Constantin 3700.
```

```
/*
```

Id-ul dat ca parametru este 11, acesta este existent in baza de date si deci nu se genereaza exceptia no_data_found. Id-urile care se vor lua in considerare in calculele din cadrul procedurii sunt practic toate id-urile incepand de la 11 inclusiv.

Se afiseaza id-ul dat(11), id-ul maxim(66), si numerele fibonacci mai mici +1 decat 66

In acest caz, din id-urile eligibile avem ca:

11, 44 si 66 sunt id-uri normale(nu sunt fibonacci)

22 este egal prin marja de eroare +1 cu 21 care este numar fibonacci

33 este egal prin marja de eroare -1 cu 34 care este numar fibonacci

55 este numar fibonacci

Avem 3 id-uri fibonacci si 3 id-uri normale. Prin urmare, nu avem mai mult de jumata + 1 din id-urile eligibile ca fiind fibonacci si deci nu se arunca exceptia too_many_fibonacci. De asemenea, nu se arunca nici exceptia no_fibonacci_found, intrucat am gasit 3 id-uri fibonacci.

Procedura se va executa normal, fara aruncare de exceptii. Pentru fiecare id fibonacci se afiseaza detaliile studentului precum si profesorii acestuia.

*/

```
execute exercitiul_6(33);
```

```
-- rezultat bun
```

The screenshot shows the Oracle SQL Developer interface. In the top-left pane, there is a list of scripts: ex6.sql, ex7.sql, ex8.sql, ex9.sql, ex10.sql, ex11.sql, ex12.sql, ex13.sql, and ex14.3.sql. The script 'ex14.3.sql' is currently open. It contains the following SQL code:

```
238 select * from student;
239
240 select p.cod_angajat, nume, prenume, salariu
241   from angajat a join profesor p on (a.cod_angajat = p.cod_angajat);
242
243 select * from participa;
244
245 execute exercitiul_6(11);
246 -- rezultat bun
247 execute exercitiul_6(33);
248 -- rezultat bun
249 execute exercitiul_6(44);
```

In the bottom-left pane, the 'Script Output' tab is selected, showing the results of the executed statements. The output includes:

```
ID-ul dat: 33
ID-ul maxim: 66

Numerele fibonacci mai mici ca 66:
1 1 2 3 5 8 13 21 34 55

Nr linie: 3
ID: 33
Nume Student: Georgescu Nicolae
Profesorii: Stanescu Constantin 3700, Stefanescu Teodora 2500.

Nr linie: 5
ID: 55
Nume Student: Florentin Daniel
Profesorii: Bogdan Octavian 5000, Stanescu Constantin 3700.
```

The 'Query Result' tab shows a table named 'STUDENT' with the following data:

COD_STUDENT	NUME	PRENUME	DATA_NASTERE	COD_SPECIALIZARE	COD_CAMPUS
1	Ilinicescu	Stefan	05-JAN-02	1	1
2	Iliescu	Mihail	12-DEC-89	2	2
3	Georgescu	Nicolae	23-MAY-95	3	3
4	Emil	Luca	28-APR-00	4	4
5	Florentin	Daniel	19-SEP-92	5	5
6	c	c	26-DEC-21	1	1

```
/*
```

Similar exemplului anterior, de data asta avem ca id dat ca parametru pe 33.

Id-urile ce vin dupa 33 sunt 44, 55 si 66. In acest caz avem ca:

33 este egal prin marja de eroare +1 cu 34 care este numar fibonacci

44 nu este numar fibonacci

55 este numar fibonacci

66 nu este numar fibonacci

Avem 2 id-uri fibonacci si 2 id-uri care nu sunt fibonacci, deci nu intram pe exceptia `too_many_fibonacci`. Nu intram nici pe exceptiile `no_data_found` sau `no_fibonacci_found`.

Pentru id-urile 33 si 55 se afiseaza informatiile corespunzatoare, intrucat acestea sunt egale +-1 fibonacci.

```
*/
```

```
execute exercitiul_6(44);
```

-- rezultat bun

The screenshot shows the Oracle SQL Developer interface. In the central workspace, the code for the stored procedure is visible:

```
244
245 execute exercitiul_6(11);
246 -- rezultat bun
247 execute exercitiul_6(33);
248 -- rezultat bun
249 execute exercitiul_6(44);
```

The line `execute exercitiul_6(44);` is highlighted. To the right, the 'Dbms Output' window displays the results of the execution:

```
ID-ul dat: 44
ID-ul maxim: 66

Numerele fibonacci mai mici ca 66:
1 1 2 3 5 8 13 21 34 55

Nr linie: 5
ID: 55
Nume Student: Florentin Daniel
Profesorii: Bogdan Octavian 5000, Stanescu Constantin 3700.
```

Below the workspace, the 'Query Result' tab is open, showing a table with student data:

COD_STUDENT	NUME	PRENUME	DATA_NASTERE	COD_SPECIALIZARE	COD_CAMPUS	
1	11	Ionescu	Stefan	05-JAN-02	1	1
2	22	Ilieșcu	Mihail	12-DEC-89	2	2
3	33	Georgescu	Nicolae	23-MAY-95	3	3
4	44	Emil	Luca	28-APR-00	4	4
5	55	Florentin	Daniel	19-SEP-92	5	5
6	66	c	c	26-DEC-21	1	1

/*

Id-ul dat ca parametru este 44, in acest caz avem ca id-uri eligibile:

44 care nu este numar fibonacci

55 care este numar fibonacci

66 care nu este numar fibonacci

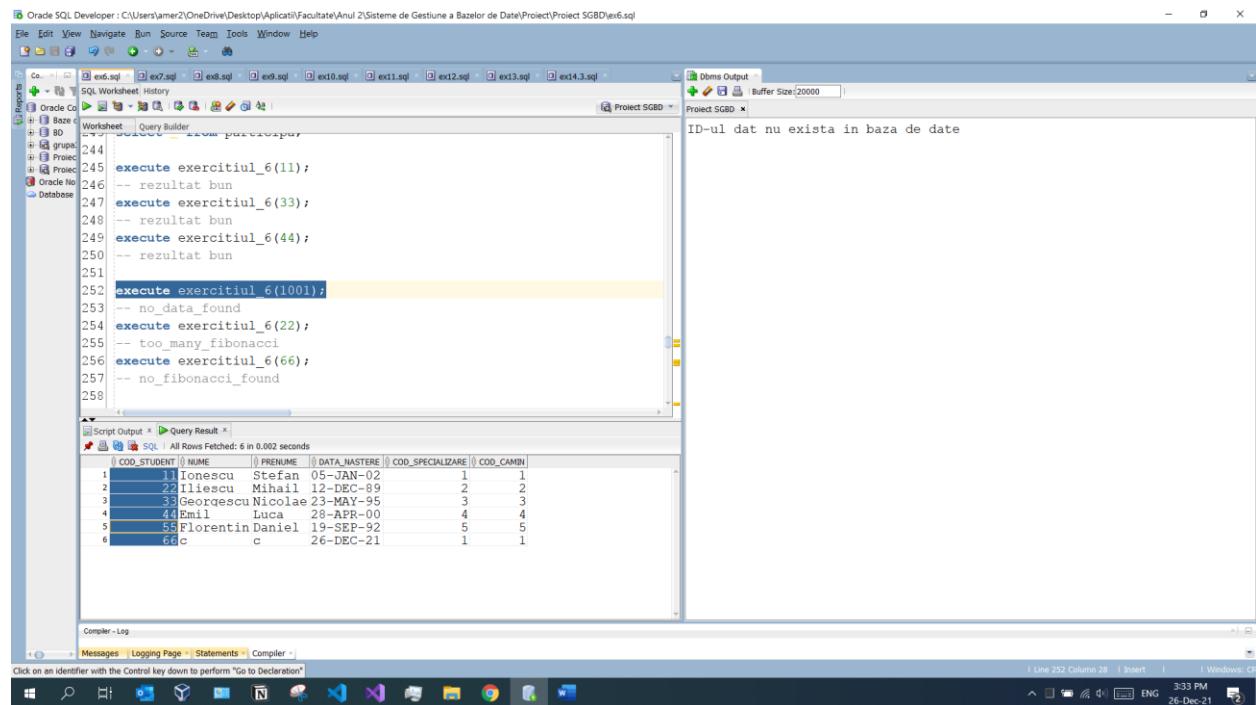
Nu mai intram pe exceptia too_many_fibonacci, intrucat avem mai putin de jumata + 1 dintre id-uri ca fiind numere fibonacci.

In acest caz, se va afisa numele studentului cu id-ul 55, precum si profesorii si salariul profesorilor la care acesta invata.

*/

```
execute exercitiul_6(1001);
```

```
-- no_data_found
```



```
244
245     execute exercitiul_6(11);
246     -- rezultat bun
247     execute exercitiul_6(33);
248     -- rezultat bun
249     execute exercitiul_6(44);
250     -- rezultat bun
251
252     execute exercitiul_6(1001);
253     -- no_data_found
254     execute exercitiul_6(22);
255     -- too many fibonacci
256     execute exercitiul_6(66);
257     -- no_fibonacci_found
258
```

ID-ul dat nu exista in baza de date

COD_STUDENT	NUME	PRENUME	DATA_NASTERE	COD_SPECIALIZARE	COD_CAMPUS	
1	11	Ionescu	Stefan	05-JAN-02	1	1
2	22	Iliescu	Mihail	12-DEC-89	2	2
3	33	Georgescu	Nicolae	23-MAY-95	3	3
4	44	Emil	Luca	28-APR-00	4	4
5	55	Florentin	Daniel	19-SEP-92	5	5
6	66	c	c	26-DEC-21	1	1

```
/*
```

Id-ul 1001 nu exista in baza de date, se arunca exceptia no_data_found. Asa cum apare in query-ul de select din poza, 1001 nu se afla printre id-urile studentilor. Atunci cand se intra pe no_data_found, se afiseaza doar un mesaj sugestiv si se opreste executia procedurii, intrucat nu se poate face nimic fara un id valid.

```
*/
```

```
execute exercitiul_6(22);
```

```
-- too_many_fibonacci
```

```
247 execute exercitiul_6(33);
248 -- rezultat bun
249 execute exercitiul_6(44);
250 -- rezultat bun
251
252 execute exercitiul_6(1001);
253 -- no data found
254 execute exercitiul_6(22);
255 -- too_many_fibonacci
256 execute exercitiul_6(66);
257 -- no_fibonacci_found
258
259 rollback;
260
261 select * from student;
```

COD_STUDENT	NUME	PRENUME	DATA_NASTERE	COD_SPECIALIZARE	COD_CAMPUS	
1	11	Ionescu	Stefan	05-JAN-02	1	1
2	22	Ilieescu	Mihail	12-DEC-89	2	2
3	33	Georgescu	Nicolae	23-MAY-95	3	3
4	44	Bm1	Luca	28-APR-00	4	4
5	55	Alionor	Florin	19-SEP-04	5	5
6	66	c	c	26-DEC-21	1	1

```
/*
```

Pentru id-ul 22 dat ca parametru, o sa avem ca id-uri fibonacci pe 22, 33 si 55, iar ca id-uri normale doar pe 44 si 66. In acest caz, mai mult de jumata + 1 dintre id-urile eligibile sunt fibonacci si se intra pe exceptia too_many_fibonacci.

```
*/
```

```
execute exercitiul_6(66);
```

```
-- no_fibonacci_found
```

```
247 execute exercitiul_6(33);
248 -- rezultat bun
249 execute exercitiul_6(44);
250 -- rezultat bun
251
252 execute exercitiul_6(1001);
253 -- no_data_found
254 execute exercitiul_6(22);
255 -- too many fibonacci
256 execute exercitiul_6(66);
257 -- no_fibonacci_found
258
259 rollback;
260
261 select * from student;
```

COD_STUDENT	NUME	PRENUME	DATA_NASTERE	COD_SPECIALIZARE	COD_CAMIN
1	11 Ionescu	Stefan	05-JAN-02	1	1
2	22 Iliescu	Mihail	12-DEC-89	2	2
3	33 Georgescu	Nicolae	23-MAY-95	3	3
4	44 Emil	Luca	28-APR-00	4	4
5	55 Florentin	Daniel	19-SEP-92	5	5
6	66 C	C	26-DEC-21	1	1

```
/*
```

Singurul id eligibil este 66, acesta fiind pe ultima linie. Cum 66 nu este egal +1 cu orice numar fibonacci, se intra pe exceptia no_fibonacci_found.

```
*/
```

```
rollback;
```

```
-- Se executa rollback pentru eliminarea inserarii facute la inceput strict pentru acest exercitiu
```

COD_STUDENT	NUME	PRENUME	DATA_NASTERE	COD_SPECIALIZARE	COD_CAMIN
1	11 Ionescu	Stefan	05-JAN-02	1	1
2	22 Iliescu	Mihail	12-DEC-89	2	2
3	33 Georgescu	Nicolae	23-MAY-95	3	3
4	44 Emil	Luca	28-APR-00	4	4
5	55 Florentin	Daniel	19-SEP-92	5	5

7.

```
/*
```

Sa se afiseze toti angajatii al caror salariu se incadreaza intre cei doi parametri a si b, precum si colegii lor de munca la facultate.

```
*/
```

```
/*
```

OBSERVATIE: Inainte de a rula urmatoarele teste este recomandat sa se verifice integritatea datelor din tabelul angajat. Mai exact, este important ca salariile sa fie aceleasi ca cele inserate initial, intrucat acestea ar putea ramane modificate de trigger-ul de la exercitiul 10.

```
*/
```

-- Cod Exercitiul 7:

```
create or replace procedure exercitiul_7(
```

```
    a  angajat.salariu%type,      -- Limita inferioara a intervalului salarial
```

```
    b  angajat.salariu%type      -- Limita superioara a intervalului salarial
```

```
)
```

```
is
```

```
    -- cursor explicit parametrizat
```

```
    -- selecteaza numele, prenumele, salariul profesorilor ca se incadreaza in intervalul salarial
```

```
    -- precum si denumirea facultatii la care acesti profesori predau
```

```
cursor c(x angajat.salariu%type, y angajat.salariu%type) is
```

```
    select nume, prenume, salariu, denumire
```

```
    from angajat join facultate using(cod_facultate)
```

```
    where salariu between x and y;
```

```
    nume      angajat.nume%type;      -- variabila pentru numele profesorului
```

```
prenume    angajat.prenume%type;    -- variabila pentru prenumele profesorului
salariu    angajat.salariu%type;    -- variabila pentru salariul profesorului
fac      facultate.denumire%type;   -- variabila pentru denumirea facultatii la care predă
profesorul
```

```
nr_linii      number := 0;  -- numarul de iteratii ale cursorului c
salariu_inexistent exception; -- exceptie pentru cazul in care nu exista salariu din
intervalul [a, b]
```

```
begin
```

```
open c(a, b);          -- Deschidem cursorul c pentru parametri a si b
loop
    fetch c into nume, prenume, salariu, fac;    -- Asignam valorile liniei curente
    variablelor corespunzatoare
    exit when c%notfound;    -- Folosim atributul notfound pentru a verifica daca am
    parcurs toate liniile tabelului. In caz afirmativ iesim din bucla
    nr_linii := nr_linii + 1; -- La fiecare iteratie a cursorului, incrementam variabila nr_linii
end loop;
close c;           -- Dupa ce am iterat prin toate liniile tabelului inchidem cursorul

if nr_linii = 0 then        -- Daca cursorul c nu a gasit nicio linie din tabel, inseamna ca nu
exista un salariu din intervalul [a, b]
    raise salariu_inexistent; -- Prin urmare, se arunca exceptia "salariu_inexistent"
end if;

-- Daca nu s-a aruncat exceptia "salariu_inexistent"
open c(a, b);          -- Putem incepe afisarea tuturor angajatilor cu salariul din intervalul [a,
b]
```

```
loop
```

```
    fetch c into nume, prenume, salariu, fac;      -- Extragem informatiile angajatului curent  
    exit when c%notfound;      -- atribut notfound
```

```
        dbms_output.put_line('Nume Angajat: ' || nume || '' || prenume); -- Afisam  
        informatiile angajatului
```

```
        dbms_output.put_line('Salariu: ' || salariu);
```

```
        dbms_output.put_line('Facultate: ' || fac);
```

```
        dbms_output.put('Colegi: ');
```

```
-- Obtainem lista colegilor angajatului curent folosindu-ne de un ciclu cursor in bucla for
```

```
for i in (
```

```
    select nume, prenume
```

```
    from angajat join facultate using(cod_facultate)      -- ciclu cursor
```

```
    where denumire = fac
```

```
) loop
```

```
    dbms_output.put(i.nume || '' || i.prenume || ',');      -- Afisam lista tuturor colegilor  
    angajatului curent
```

```
end loop;
```

```
dbms_output.new_line; -- Afisam new_line pentru a se afisa lista anterior mentionata
```

```
dbms_output.new_line; -- Afisam inca 2 new_line-uri pentru spatiere
```

```
dbms_output.new_line;
```

```
end loop;
```

```
close c; -- Inchidem cursorul c

exception

-- nu intalnim cazul in care sa avem eroarea no_data_found

-- intrucat in cazul cursoarelor, chiar daca nu se gaseste nicio linie in urma

-- select-urilor, cursoarele raman valide, dar goale

-- in cazul care cursoarelor goale, avem exceptia "salariu_inexistent"

-- when no_data_found then dbms_output.put_line('Nu au fost gasite date in baza de date');

-- nu intalnim cazul in care sa avem eroarea too_many_rows

-- intrucat toate select-urile facute sunt pentru cursoare

-- when too_many_rows then dbms_output.put_line('Too Many Rows');

when salariu_inexistent then dbms_output.put_line('Nu exista niciun salariu din intervalul ['
|| a || ',' || b || ']');

when others then dbms_output.put_line('Alt tip de eroare!');

end exercitiul_7;
/
```

-- Rezultate/Explicatii:

```
/*
```

Pentru a vedea mai bine cum se executa programul, vom selecta tabelul tuturor angajatilor si vom sorta dupa salariu.

```
*/
```

```
select * from angajat order by salariu;
```

	COD_ANGAJAT	NUME	PRENUME	DATA_NASTERE	SALARIU	COD_FACULTATE
1	1	Popescu	Ion	17-JAN-87	1500	10
2	4	Popovici	Alexandra	21-AUG-82	1800	20
3	8	Mihai	George	27-JUL-82	1900	30
4	5	Ionescu	Mihai	12-SEP-96	2300	20
5	2	Stefanescu	Teodora	25-DEC-86	2500	10
6	11	Alexandrescu	Marius	15-FEB-81	2500	40
7	7	Popa	Andra	08-APR-90	3200	30
8	6	Stanescu	Constantin	19-OCT-89	3700	20
9	3	Mihaleascu	Andrei	06-MAY-92	3800	10
10	9	Marinescu	Elena	19-SEP-77	4100	30
11	10	Boqdan	Octavian	04-NOV-78	5000	40

```
/*
```

De asemenea, vom selecta informatiile din tabelul facultate.

```
*/
```

```
select * from facultate;
```

	COD_FACULTATE	DENUMIRE	RANKING	COD_LOCATIE
1	10	Facultatea de Matematica si Informatica	1	2
2	20	Facultatea de Automatica si Calculatoare	2	5
3	30	Facultatea de Agronomie	3	1
4	40	Facultatea de Medicina	4	3
5	50	Facultatea de Constructii	5	4

```
execute exercitiul_7(3800, 5000);
```

-- rezultat bun

The screenshot shows the Oracle SQL Developer interface. On the left, the code editor displays a PL/SQL block:

```
92: */
93: *
94:
95:
96:
97: execute exercitiul_7(3800, 5000);
98: -- rezultat bun
99: execute exercitiul_7(100000, 1000001);
100: -- salariu_inexistent
101:
102: select * from angajat order by salariu;
103:
```

The output window on the right shows the results of the execution:

```
Nume Angajat: Mihailescu Andrei
Salariu: 3800
Facultate: Facultatea de Matematica si Informatica
Colegi: Popescu Ion, Stefanescu Teodora, Mihai George, Popa Andra,
```

```
Nume Angajat: Marinescu Elena
Salariu: 4100
Facultate: Facultatea de Agronomie
Colegi: Bogdan Octavian, Mihai George,
```

```
Nume Angajat: Bogdan Octavian
Salariu: 5000
Facultate: Facultatea de Medicina
Colegi: Alexandrescu Marius,
```

The results table shows 11 rows of data:

COD_ANGAJAT	NUME	PRENUME	DATA_NASTERE	SALARU	COD_FACULTATE
1	Popescu	Ion	17-JAN-87	1500	10
2	Popovici	Alexandra	21-AUG-82	1800	20
3	Mihai	George	27-JUL-82	1900	30
4	Ionescu	Mihai	12-SEP-96	2300	20
5	Stefanescu	Teodora	25-DEC-86	2500	10
6	Alexandrescu	Marius	15-FEB-81	2500	40
7	Popa	Andra	08-APR-90	3200	30
8	Stanescu	Constantin	19-OCT-89	3700	20
9	Mihailescu	Andrei	06-MAY-92	3800	10
10	Marinescu	Elena	19-SEP-77	4100	30
11	Bogdan	Octavian	04-NOV-78	5000	40

/*

Asa cum reiese din tabelul angajatilor sortati dupa salariu, pentru intervalul salarial [3800, 5000] avem ultimii 3 angajati din tabel, anume: Mihailescu Andrei, Marinescu Elena, Bogdan Octavian.

- Mihailescu Andrei lucreaza la facultatea cu codul 10, anume Facultatea de Matematica si Informatica, alaturi de el lucreaza si Popescu Ion si Stefanescu Teodora.
- Marinescu Elena lucreaza la facultatea cu codul 30, anume Facultatea de Agronomie, alaturi de ea lucreaza Mihai George si Popa Andra.
- Bogdan Octavian lucreaza la facultatea cu codul 40, anume Facultatea de Medicina, alaturi de el lucreaza Alexandrescu Marius.

*/

```
execute exercitiul_7(100000, 1000001);
```

```
-- salariu_inexistent
```

The screenshot shows the Oracle SQL Developer interface. In the top-left pane, there are several tabs: ex7.sql, ex8.sql, ex9.sql, ex10.sql, ex11.sql, ex12.sql, ex13.sql, and ex14.3.sql. The current tab is 'ex11.sql'. The code in the editor is:

```
94
95
96
97 execute exercitiul_7(3800, 5000);
98 -- rezultat bun
99 execute exercitiul_7(100000, 1000001);
100 -- salariu_inexistent
101
102 select * from angajat order by salariu;
103
104 select * from facultate;
```

In the top-right pane, the 'Dbms Output' window displays the message: "Nu exista niciun salariu din intervalul [100000, 1000001]". Below the code editor, the 'Script Output' and 'Query Result' panes are visible. The 'Query Result' pane shows a table with 11 rows of data:

COD_ANGAJAT	NUME	PRENUME	DATA_NASTERE	SALARU	COD_FACULTATE
1	Popescu	Ion	17-JAN-87	1500	10
2	Popovici	Alexandra	21-AUG-82	1800	20
3	Mihai	George	27-JUL-82	1900	30
4	Ionescu	Mihai	12-SEP-96	2300	20
5	Stefanescu	Teodora	25-DEC-86	2500	10
6	Alexandrescu	Marius	15-FEB-81	2500	40
7	Popa	Andra	08-APR-90	3200	30
8	Stanescu	Constantin	19-OCT-89	3700	20
9	Mihaleascu	Andrei	06-MAY-92	3800	10
10	Marinescu	Elena	19-SEP-77	4100	30
11	Bodan	Octavian	04-NOV-78	5000	40

At the bottom of the interface, the status bar shows: Line 99 Column 1 | Insert | Windows: C:\ | 4:31 PM | ENG | 26-Dec-21.

```
/*
```

Asa cum rezulta din tabelul angajatilor, nu exista salarii din intervalul [100000, 1000001]. Prin urmare, se intra pe exceptia "salariu_inexistent" si se afiseaza un mesaj sugestiv.

```
*/
```


8.

```
/*
```

Pentru prenumele si numele unui student date ca parametru se va returna numele tarii in care se afla facultatea la care este inscris studentul.

In cazul in care studentul este inscris la mai multe facultati, se va alege facultatea cu ranking-ul mai mare.

De asemenea, se vor afisa in dbms_output numele specializarii, numele caminului in care este cazat studentul precum si numele tarii in care se afla facultatea.

Propozitia afisata va avea forma:

Studentul <><> urmeaza specializarea <><> dintr-o facultate din <><>,
fiind cazat in caminul <><>.

```
*/
```

-- Cod Exercitiul 8:

```
create or replace function exercitiul_8(
    nume_student      in  student.nume%type,          -- Numele studentului dat ca parametru
    de intrare de tip IN

    prenume_student   in  student.prenume%type        -- Prenumele studentului dat ca
    parametru de intrare de tip IN

)
return tara.nume%type      -- Tipul de data returnat este numele tarii in care se afla
facultatea de ranking maxim a studentului dat ca parametru

is

    cod              student.cod_student%type;      -- Codul studentului dat ca parametru
    nume_tara         tara.nume%type;                -- Numele tarii in care afla facultatea
    studentului

    nume_specializare specializare.denumire%type;  -- Numele specializarii la care este
    inscris studentul

    nume_camin       camin.denumire%type;           -- Numele caminului studentului
```

```

begin

    -- putem avea erorile no_data_found sau too_many_rows

    -- no_data_found atunci cand nu exista un student cu numele si prenumele dat

    -- too_many_rows atunci cand avem cel putin doi studenti cu nume si prenume identice

    select cod_student    -- selectam codul studentului cu numele si prenumele date

    into cod

    from student

    where initcap(nume) = initcap(nume_student) and initcap(prenume) =
initcap(prenume_student);

    -- Selectarea se face fara case sensitivity, numele studentului putand fii scris cu numere mici
    sau mari.

    -- Daca nu s-a aruncat nicio exceptie, atunci putem cauta numele caminului, numele
    specializarii si numele tarii

    -- Din moment ce selectam informatii din minim 3 tabele diferite,
    -- avem de facut join-uri care sa conecteze informatiile astfel incat
    -- sa se pastreze corectitudinea datelor.

select c.denumire "Camin", sp.denumire "Specializare", t.nume "Tara"
into nume_camin, nume_specializare, nume_tara
from student s
join camin c on (s.cod_camin = c.cod_camin)
join specializare sp on (s.cod_specializare = sp.cod_specializare)
join studiaza st on (s.cod_student = st.cod_student)
join facultate f on (f.cod_facultate = st.cod_facultate)
join locatie l on (l.cod_locatie = f.cod_locatie)
join tara t on (l.cod_tara = t.cod_tara)
where f.rank = (

```

```

select min(ranking) -- cu cat rankingul este mai mic cu atat facultatea este mai de top
from student s2
    join studiaza st2 on (s2.cod_student = st2.cod_student)
    join facultate f2 on (f2.cod_facultate = st2.cod_facultate)
    where s2.cod_student = cod -- Selectam ranking-ul maxim dintre toate facultatiile
urmate de studentul dat ca parametru
) and s.cod_student = cod; -- Selectam detaliiile despre facultatea al carei ranking este egal cu
ranking-ul maxim dintre toate facultatile urmatoare de studentul dat ca parametru

-- Afisam detaliiile obtinute
dbms_output.put('Studentul ' || nume_student || ' ' || prenume_student);
dbms_output.put(' urmeaza specializarea ' || nume_specializare);
dbms_output.put(' dintr-o facultate din ' || nume_tara || ',');
if nume_camin = null then -- Daca studentul nu este cazat la vreun camin, afisam un mesaj
sugestiv
    dbms_output.put(' studentul nefiind cazat la vreun camin.');
else
    dbms_output.put(' fiind cazat in caminul ' || nume_camin || '.');
end if;

dbms_output.new_line; -- Pentru dbms_output.put();

return(nume_tara); -- Functia returneaza numele tarii in care se afla facultatea de ranking
maxim urmatoare de student

exception
    when no_data_found then -- Atunci cand nu exista studentul dat ca parametru in baza
de date

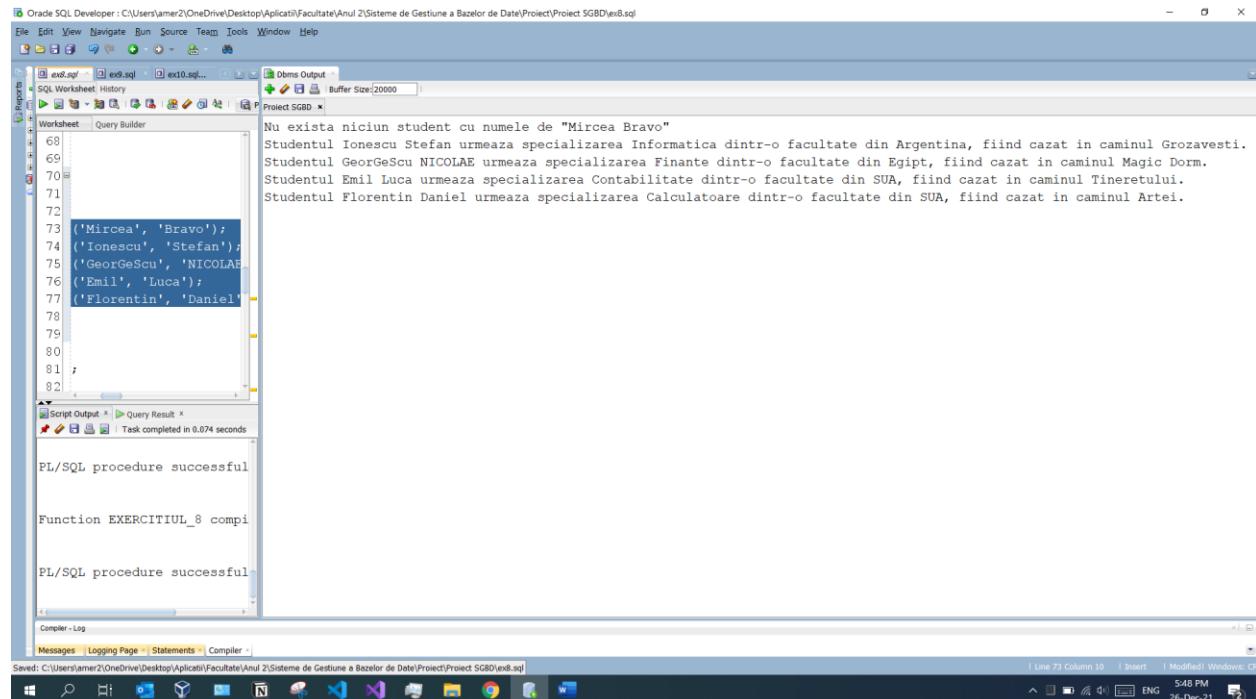
```

```
    dbms_output.put_line('Nu exista niciun student cu numele de "' || nume_student || ' ' ||  
    prenume_student || ''');  
  
    return();          -- Returnam sirul vid, intrucat nu putem afla tara in care se afla  
    facultatea  
  
    when too_many_rows then      -- Exista mai multi studenti cu acelasi nume si prenume  
  
        dbms_output.put_line('Exista mai multi studenti cu numele de "' || nume_student || ' ' ||  
        prenume_student || ''');  
  
        return();  
  
    when others then           -- Orice alt tip de eroare  
  
        dbms_output.put_line('Alt tip de eroare!');  
  
        return();  
  
end exercitiul_8;  
  
/
```

-- Rezultate/Explicatii:

```
declare  
    a tara.nume%type;  
  
begin  
    a := exercitiul_8('Mircea', 'Bravo');  
  
    a := exercitiul_8('Ionescu', 'Stefan');  
  
    a := exercitiul_8('GeorGeScu', 'NICOLAE');  
  
    a := exercitiul_8('Emil', 'Luca');  
  
    a := exercitiul_8('Florentin', 'Daniel');  
  
end;
```

```
/
```



```
Nu exista niciun student cu numele de "Mircea Bravo"  
Studentul Ionescu Stefan urmeaza specializarea Informatica dintr-o facultate din Argentina, fiind cazat in caminul Grozavesti.  
Studentul GeorGeScu NICOLAE urmeaza specializarea Finante dintr-o facultate din Egipt, fiind cazat in caminul Magic Dorm.  
Studentul Emil Luca urmeaza specializarea Contabilitate dintr-o facultate din SUA, fiind cazat in caminul Tineretului.  
Studentul Florentin Daniel urmeaza specializarea Calculatoare dintr-o facultate din SUA, fiind cazat in caminul Artei.
```

```
PL/SQL procedure successful  
  
Function EXERCITIUL_8 compiled  
  
PL/SQL procedure successful
```

```
/*
```

Studentul “Mircea Bravo” nu se regaseste in tabelul student, “GeorGeScu NICOLAE” este de fapt “Georgescu Nicolae” si se regaseste in tabelul student. Restul studentilor se regasesc in tabelul student si deci sunt afisati corespunzator.

```
*/
```


9.

```
/*
```

Pentru un ranking al unei facultati dat ca parametru, sa se returneze numele facultatii, impreuna cu nota cea mai mare obtinuta in cadrul facultatii, numele disciplinei, numele studentului si numele profesorului coordonator.

Daca exista mai multi studenti cu nota maxima, se va afisa doar primul in ordine alfabetica dupa nume si prenume.

```
*/
```

-- Cod Exercitiul 9:

```
create or replace procedure exercitiul_9(
```

```
    ranking_facultate facultate.ranking%type      -- ranking-ul facultatii dat ca parametru
)
```

```
is
```

```
    cod          facultate.cod_facultate%type;    -- Codul facultatii
    nume_facultate   facultate.denumire%type;      -- Numele facultatii
    nota_maxima     examinare.nota%type;           -- Nota maxima obtinuta in cadrul facultatii
    nume_disciplina  disciplina.denumire%type;      -- Numele disciplinei asupra carei s-a
    obtinut nota maxima

    nume_student     student.nume%type;            -- Numele studentului care a obtinut nota
    maxima la disciplina din cadrul facultatii

    nume_profesor    angajat.nume%type;           -- Numele profesorului coordonator
    disciplinei cu nota maxima
```

```
begin
```

```
-- verificam daca codul exista
```

```
select cod_facultate
```

```

into cod
from facultate
where ranking = ranking_facultate;
-- Daca codul nu ar fi exista, s-ar fi generat eroare no_data_found si s-ar fi oprit executia
procedurii

select * -- Selectam informatiile necesare din tabele multiple folosind join-uri
into nume_facultate, nota_maxima, nume_disciplina, nume_student, nume_profesor
from(
    select f.denumire "Facultate",
        e.nota "Nota",
        d.denumire "Disciplina",
        s.nume || ' ' || s.prenume "Student", -- Numele studentului si profesorului sunt
complete
        a.nume || ' ' || a.prenume "Profesor"

    from student s
        join participa p on (p.cod_student = s.cod_student)
        join examinare e on (e.cod_disciplina = p.cod_disciplina)
        join angajat a on (a.cod_angajat = p.cod_angajat)
        join profesor pf on (a.cod_angajat = pf.cod_angajat)
        join disciplina d on (p.cod_disciplina = d.cod_disciplina)
        join programa pr on (d.cod_disciplina = pr.cod_disciplina)
        join specializare sp on (pr.cod_specializare = sp.cod_specializare)
        join facultate f on (f.cod_facultate = sp.cod_facultate)
    where nota = ( -- Selectam toate detaliile pentru disciplina cu nota maxima
        select max(nota) -- Subcerere in care electam nota maxima din cadrul facultatii date
        from examinare e
            join disciplina d on (e.cod_disciplina = d.cod_disciplina)

```

```
join programa p on (d.cod_disciplina = p.cod_disciplina)
join specializare sp on (sp.cod_specializare = p.cod_specializare)
join facultate f on (sp.cod_facultate = f.cod_facultate)
where f.cod_facultate = cod
) and f.cod_facultate = cod
order by s.nume, s.prenume -- Ordonam selectia dupa nume si prenume
)

where rownum = 1; -- In cazul in care exista mai multi studenti, selectam doar primul
ordonat dupa nume si prenume
```

-- Afisam informatiile obtinute

```
dbms_output.put_line('Facultate: ' || nume_facultate);
dbms_output.put_line('Nota: ' || nota_maxima);
dbms_output.put_line('Disciplina: ' || nume_disciplina);
dbms_output.put_line('Student: ' || nume_student);
dbms_output.put_line('Profesor: ' || nume_profesor);
```

exception

```
when no_data_found then dbms_output.put_line('Nu exista facultate cu ranking-ul dat');
when too_many_rows then dbms_output.put_line('Exista mai multe facultati care au
ranking-ul egal cu ' || ranking_facultate);
when others then dbms_output.put_line('Alt tip de eroare!');

end exercitiul_9;
/
```

-- Rezultate/Explicatii:

select * from facultate;

	COD_FACULTATE	DENUMIRE	RANKING	COD_LOCATIE
1		10 Facultatea de Matematica si Informatica	1	2
2		20 Facultatea de Automatica si Calculatoare	2	5
3		30 Facultatea de Agronomie	3	1
4		40 Facultatea de Medicina	4	3
5		50 Facultatea de Constructii	5	4

execute exercitiul_9(1);

execute exercitiul_9(2);

-- Rezultate Corecte

The screenshot shows the Oracle SQL Developer interface. In the top right, the 'Dbsms Output' window displays the results of the PL/SQL code execution:

```
Facultate: Facultatea de Matematica si Informatica
Nota: 10
Disciplina: Matematica
Student: Georgescu Nicolae
Profesor: Stefanescu Teodora

Facultate: Facultatea de Automatica si Calculatoare
Nota: 9.5
Disciplina: Chimie
Student: Emil Luca
Profesor: Popovici Alexandra
```

In the bottom left, the 'Query Result' window shows the expected result of the SELECT query:

	COD_FACULTATE	DENUMIRE	RANKING	COD_LOCATIE
1		10 Facultatea de Matematica si Informatica	1	2
2		20 Facultatea de Automatica si Calculatoare	2	5
3		30 Facultatea de Agronomie	3	1
4		40 Facultatea de Medicina	4	3
5		50 Facultatea de Constructii	5	4

```
execute exercitiul_9(10);
```

```
-- no_data_found
```

The screenshot shows the Oracle SQL Developer interface. The 'Worksheet' tab contains the following PL/SQL code:

```
when no_data_found then dbms_output.put_line('Nu exista facultate cu ranking-ul');
when too_many_rows then dbms_output.put_line('Există mai multe facultăți care au același ranking');
when others then dbms_output.put_line('Alt tip de eroare!');

end exercitiul_9;
/
execute exercitiul_9(1);
execute exercitiul_9(2);

execute exercitiul_9(10);
-- no_data_found
```

The 'Output' tab shows the result of the last execute statement:

```
Nu exista facultate cu ranking-ul
```

The 'Script Output' tab shows the results of the previous execute statements:

COD_FACULTATE	DENUMIRE	RANKING	COD_LOCATIE
1	10 Facultatea de Matematica si Informatica	1	2
2	20 Facultatea de Automatica si Calculatoare	2	5
3	30 Facultatea de Aeronație	3	1
4	40 Facultatea de Medicina	4	3
5	50 Facultatea de Constructii	5	4

```
/*
```

Nu exista facultate al carei ranking sa fie 10 si deci se intra pe exceptia no_data_found.

```
*/
```


10.

```
/*
```

Datorita deficitului bugetar, la inserarea unui nou angajat in tabel, sa se micsoreze salariul tuturor angajatilor cu un procent de 20% pentru a putea acomoda salariul noului angajat.

Din moment ce dorim sa ne folosim de acelasi tabel care va declansa trigger-ul vom intra in cazul de tabel mutating, caz pe care il rezolvam folosind trigger compound.

```
*/
```

-- Cod Exercitiul 10:

```
create or replace trigger exercitiul_10    -- Cream(sau inlocuim in cazul existentei) triggerul  
for insert on angajat           -- Aceasta va actiona asupra inserarilor asupra tablei angajat  
compound trigger                -- Trigger de tip Compound
```

```
before statement is           -- La declansarea trigger-ului, inainte de inserare se va afisa un  
mesaj sugestiv
```

```
begin  
    dbms_output.put_line('A fost apelat trigger-ul compound');  
end before statement;
```

```
after statement is           -- Dupa ce s-a inserat angajatul, se va micsora salariul fiecarui  
angajat
```

```
begin  
    update angajat  
        set salariu = salariu - 0.2*salariu;  -- Nu exista clausa where, intrucat dorim sa modificam  
                                         salariul tuturor angajatilor  
    end after statement;
```

```
end exercitiul_10;
```

```
/
```

-- Rezultate/Explicatii:

```
/*
```

Tabelul angajatilor inainte de orice inserare

```
*/
```

```
select * from angajat;
```

	COD_ANGAJAT	NUME	PRENUME	DATA_NASTERE	SALARIU	COD_FACULTATE
1	1	Popescu	Ion	17-JAN-87	1500	10
2	2	Stefanescu	Teodora	25-DEC-86	2500	10
3	3	Mihailescu	Andrei	06-MAY-92	3800	10
4	4	Popovici	Alexandra	21-AUG-82	1800	20
5	5	Ionescu	Mihai	12-SEP-96	2300	20
6	6	Stanescu	Constantin	19-OCT-89	3700	20
7	7	Popa	Andra	08-APR-90	3200	30
8	8	Mihai	George	27-JUL-82	1900	30
9	9	Marinescu	Elena	19-SEP-77	4100	30
10	10	Boqdan	Octavian	04-NOV-78	5000	40
11	11	Alexandrescu	Marius	15-FEB-81	2500	40

-- In cazul in care inca era activat trigger-ul de la exercitiul 11, il dezactivam

```
alter trigger nivel_linie disable;
```

```
select avg(salariu) from angajat;
```

-- salariul mediu este aproximativ 2936

```
select avg(salariu - salariu*0.2) from angajat;
```

-- salariul mediu calculat din toate salariile existente

-- inainte de inserare reduse cu 20% este aproximativ 2349

```
insert into angajat  
values (13, 'c', 'c', sysdate, 3000, 10);
```

-- Inseram o valoare oarecare in tabelul angajat pentru a declansa trigger-ul

```
Oracle SQL Developer : C:\Users\lamer2\OneDrive\Desktop\Aplicati\Facultate\Anul 2\Sisteme de Gestie a Bazelor de Date\Project SGBD\ex10.sql  
File Edit View Navigate Run Source Team Tool Window Help  
ext10.sql ext11.sql ext2.sql ext13.sql ex14.3.sql  
SQL Worksheet History  
Worksheet Query Builder  
32 select avg(salariu) from angajat;  
33 -- salariul mediu este aproximativ 2936  
34  
35 select avg(salariu - salariu*0.2) from angajat;  
36 -- salariul mediu calculat din toate salariile existente  
37 -- inainte de inserare reduse cu 20% este aproximativ 2349  
38  
39 insert into angajat  
40 values (13, 'c', 'c', sysdate, 3000, 10);  
41 -- Inseram o valoare oarecare in tabelul angajat pentru a declansa trigger-ul  
42  
43 select avg(salariu)  
44 from angajat  
45 where cod_angajat <> 13;  
46 -- dupa inserarea angajatului 13, toate salariile existente  
47 -- inainte de inserare s-au micsorat cu 20%, prin urmare salariul  
48 -- mediu s-a micsorat cu 20%  
A fost apelat trigger-ul compound  
Script Output x Query Result x  
Task completed in 0.09 seconds  
Rollback complete.  
Trigger EXERCITIUL 10 compiled.  
1 row inserted.
```

```
select avg(salariu)  
from angajat  
where cod_angajat <> 13;  
-- dupa inserarea angajatului 13, toate salariile existente  
-- inainte de inserare s-au micsorat cu 20%, prin urmare salariul  
-- mediu s-a micsorat cu 20%
```

Oracle SQL Developer : C:\Users\lamer2\OneDrive\Desktop\Aplicati\Facultate\Anul 2\Sisteme de Gestie a Bazelor de Date\Project\Project SGBD\ex10.sql

File Edit View Navigate Run Source Team Tools Window Help

SQL Worksheet History

Worksheet Query Builder

```

35 select avg(salariu) = 'Salariu*0.2' from angajat;
36 -- salariul mediu calculat din toate salariile existente
37 -- inainte de inserare reduse cu 20% este aproximativ 2349
38
39 insert into angajat
40 values (13, 'c', 'c', sysdate, 3000, 10);
41 -- inseram o valoare oarecare in tabelul angajat pentru a declansca trigger-ul
42
43 select avg(salariu)
44 from angajat
45 where cod_angajat < 13;
46-- dupa inserarea angajatului 13, toate salariile existente
47-- inainte de inserare s-au micsorat cu 20%, prin urmare salariul
48-- mediu s-a micsorat cu 20%
49
50 select * from angajat;
51 rollback;

```

Script Output x | Query Result x

All Rows Fetched: 1 in 0.007 seconds

	cod_angajat	nume	prenume	data_nasterere	salariu	cod_facultate
1	1	Popescu	Ion	17-JAN-87	1200	10
2	2	Stefanescu	Theodora	25-DEC-86	2000	10
3	3	Mihai	Andrei	06-MAR-90	2000	10
4	4	Papocici	Alexandra	21-AUG-82	1440	20
5	5	Ionescu	Mihaianda	12-SEP-96	1840	20
6	6	Stanescu	Constantin	19-OCT-89	2960	20
7	7	Popa	Andra	08-APR-90	2560	30
8	8	Mihai	George	27-JUL-82	1520	30
9	9	Marinescu	Elena	19-SEP-77	3280	30
10	10	Bogdan	Octavian	04-NOV-78	4000	40
11	11	Alexandrescu	Marius	15-FEB-81	2000	40
12	13	c	c	26-DEC-21	2400	10

Compiler Log messages warning notes diagnostics | Compiler

Click on an identifier with the Control key down to perform "Go to Declaration"

Line 48 Column 29 | Insert | Modified | Windows: CP

6:44 PM ENG 26-Dec-21

select * from angajat;

-- Putem observa ca toate salariile s-au micsorat cu 20%

Oracle SQL Developer : C:\Users\lamer2\OneDrive\Desktop\Aplicati\Facultate\Anul 2\Sisteme de Gestie a Bazelor de Date\Project\Project SGBD\ex10.sql

File Edit View Navigate Run Source Team Tools Window Help

SQL Worksheet History

Worksheet Query Builder

```

43 select avg(salariu)
44 from angajat
45 where cod_angajat < 13;
46-- dupa inserarea angajatului 13, toate salariile existente
47-- inainte de inserare s-au micsorat cu 20%, prin urmare salariul
48-- mediu s-a micsorat cu 20%
49
50 select * from angajat;
51 -- Putem observa ca toate salariile s-au micsorat cu 20%
52
53 rollback;
54
55 delete from angajat
56 where cod_angajat = 13;

```

Script Output x | Query Result x | Query Result 1 x

All Rows Fetched: 12 in 0.004 seconds

	cod_angajat	nume	prenume	data_nasterere	salariu	cod_facultate
1	1	Popescu	Ion	17-JAN-87	1200	10
2	2	Stefanescu	Theodora	25-DEC-86	2000	10
3	3	Mihai	Andrei	06-MAR-90	2000	10
4	4	Papocici	Alexandra	21-AUG-82	1440	20
5	5	Ionescu	Mihaianda	12-SEP-96	1840	20
6	6	Stanescu	Constantin	19-OCT-89	2960	20
7	7	Popa	Andra	08-APR-90	2560	30
8	8	Mihai	George	27-JUL-82	1520	30
9	9	Marinescu	Elena	19-SEP-77	3280	30
10	10	Bogdan	Octavian	04-NOV-78	4000	40
11	11	Alexandrescu	Marius	15-FEB-81	2000	40
12	13	c	c	26-DEC-21	2400	10

Compiler Log messages warning notes diagnostics | Compiler

Click on an identifier with the Control key down to perform "Go to Declaration"

Line 50 Column 23 | Insert | Modified | Windows: CP

6:47 PM ENG 26-Dec-21

-- Odata ce am rulat codul, putem realiza un rollback pentru

-- A modifica la loc salariile si a sterge angajatul inserat
rollback;

-- Putem face disable la trigger

alter trigger exercitiul_10 disable;

11.

```
/*
```

Daca se doreste inserarea unui nou angajat, acesta va putea fi inserat doar daca salariul lui adaugat cu salariul tuturor angajatilor nu depaseste valoarea de 37000.

Daca conditia nu este indeplinita atunci se vor genera erori diferite in felul urmator:

- Daca salariul este inserat(insert) si este mai mare decat salariul minim, se va afisa un mesaj, daca este mai mic, se va afisa alt mesaj
- Daca salariul este modificat(update) si este mai mare decat salariul mediu, se va afisa un mesaj, daca este mai mic, se va afisa alt mesaj

Salariul minim si cel mediu vor fi calculate din salariile tuturor angajatilor deja existenti in baza de date, cu exceptia salariului angajatului care declanseaza triggerul.

In orice caz, daca triggerul va fi declansat acesta va apela o eroare.

Din moment ce se va lucra pe tabel mutating se va crea un pachet cu toate informatiile necesare din tabelul respectiv. Pachetul va fi populat prin intermediul unui trigger la nivel de comanda folosind precedenta triggerilor before la nivel de comanda vs la nivel de linie

```
*/
```

-- Cod Exercitiul 11:

```
create or replace package informatii_angajat      -- Cream pachetul ce va contine toate  
informatiile pe care le vrem din tabela mutating
```

```
is
```

```
    salariu_minim    angajat.salariu%type;    -- Salariul minim din tabelul angajat  
    salariu_mediul   angajat.salariu%type;    -- Salariul mediu din tabelul angajat  
    suma_salarii     angajat.salariu%type;    -- Suma salariilor din tabelul angajat
```

end;
/

-- Triggerul la nivel de comanda before are o precedenta mai mare fata de triggerul la nivel de linie before

-- Prin urmare, inainte de a declansă trigger-ul la nivel de linie

-- ne putem folosi de cel de la nivel de comanda pentru a extrage informatiile de care avem nevoie

-- din tabela mutating angajat

-- Cream triggerul la nivel de comanda

before insert or update on angajat -- Acesta va actiona inainte de insert-urile sau update-urile pe tabela angajat

begin

select min(salariu), avg(salariu), sum(salariu) -- Selectam salariul minim, mediu si suma salariilor

into informatii_angajat.salariu_minim, -- In cele 3 atribute ale pachetului
informatii_angajat

informatii_angajat.salariu_mediul,

informatii_angajat.suma_salarii

from angajat;

```
-- where cod_angajat <> :old.cod_angajat;
```

end;

/

```
create or replace trigger nivel_linie -- Cream triggerul la nivel de linie
```

before insert or update on angajat -- Acesta va actiona inainte de insert-urile sau update-urile pe tabela angajat

-- numim variibala externa new ca fiind "nou"

for each row -- Trigger la nivel de linie

```

begin

    if :nou.salariu + informatii_angajat.suma_salarii > 37000 then      -- Daca salariul nou
adaugat la suma salariilor este mai mare decat 37000

        if inserting then    -- In cazul in care salariul nou este inserat

            if :nou.salariu >= informatii_angajat.salariu_minim then      -- Daca acesta este mai
mare sau egal decat salariul minim afisam un mesaj de eroare

                raise_application_error(-20001,'Salariul inserat este mai mare decat salariul minim');

            elsif :nou.salariu < informatii_angajat.salariu_minim then      -- Altfel, afisam alt mesaj
de eroare

                raise_application_error(-20001, 'Salariul inserat este mai mic decat salariul minim');

            end if;

        elsif updating then    -- In cazul in care salariul nou este updatat

            if :nou.salariu >= informatii_angajat.salariu_mediul then      -- Daca acesta este mai
mare sau egal cu salariul mediu afisam un mesaj de eroare

                raise_application_error(-20001, 'Salariul updatat este mai mare decat salariul mediu');

            elsif :nou.salariu < informatii_angajat.salariu_mediul then      -- Altfel, afisam alt mesaj
de eroare

                raise_application_error(-20001, 'Salariul updatat este mai mic decat salariul mediu');

            end if;

        end if;

    end if;

end;
/

```

-- Rezultate/Explicatii:

```
select * from angajat;
```

COD_ANGAJAT	NUME	PRENUME	DATA_NASTERE	SALARIU	COD_FACULTATE
1	Popescu	Ion	17-JAN-87	1500	10
2	Stefanescu	Teodora	25-DEC-86	2500	10
3	Mihaileascau	Andrei	06-MAY-92	3800	10
4	Popovoci	Alexandra	21-AUG-82	1800	20
5	Ionescu	Mihai	12-SEP-96	2300	20
6	Stanescu	Constantin	19-OCT-89	3700	20
7	Popa	Andra	08-APR-90	3200	30
8	Mihai	George	27-JUL-82	1900	30
9	Marinescu	Elena	19-SEP-77	4100	30
10	Boqdan	Octavian	04-NOV-78	5000	40
11	Alexandrescu	Marius	15-FEB-81	2500	40

```
select min(salariu), avg(salariu), sum(salariu)
```

from angajat;

-- Salariul minim: 1500

-- Salariul mediu: 2936(aproximativ)

-- Suma salariilor: 32300

insert into angajat

```
values (12, 'c', 'c', sysdate, 3000, 10);
```

-- inserarea se face cu succes, salariul 3000 nu afecteaza conditia impusa

Oracle SQL Developer : C:\Users\lamer2\OneDrive\Desktop\Aplicati\Facultate\Anul 2\Sisteme de Gestie a Bazelor de Date\Project\Project SGBD\ex11.sql

File Edit View Navigate Run Source Team Tools Window Help

SQL Worksheet History

Reports Worksheet Query Builder

```

46 end;
47 /
48
49 select min(salariu), avg(salariu), sum(salariu)
50 from angajat;
51
52 select * from angajat;
53
54 insert into angajat
55 values (12, 'c', 'c', sysdate, 3000, 10);
-- inserarea se face cu succes, salariul 3000 nu afecteaza conditia impusa
57
58 select * from angajat;
59
60 rollback;
61
62 insert into angajat
63 values (12, 'c', 'c', sysdate, 3000000, 10);

```

Script Output x Query Result x

All Rows Fetched: 12 in 0.001 seconds

COD_ANGAJAT	NUME	PRENUME	DATA_NASTERE	SALARIU	COD_FACULTATE
4	Popovici	Alexandra	21-AUG-82	1800	20
5	Ionescu	Mihai	12-SEP-96	2300	20
6	Stanescu	Constantin	19-JUN-00	3100	20
7	Dobrescu	Andra	27-APR-90	3200	30
8	Mihai	George	27-JUL-82	1900	30
9	Marinescu	Elena	19-SEP-77	4100	30
10	Bozdan	Octavian	04-NOV-78	5000	40
11	Alexandrescu	Marius	15-FEB-81	2500	40
12	c	c	22-DEC-21	3000	10

Compiler - Log

Messages Logging Page Statements Compiler

Click on an identifier with the Control key down to perform "Go to Declaration"

Line 58 Column 23 | Insert | Modified: Windows_CP

10:36 AM ENG 22-Dec-21

rollback;

insert into angajat

values (13, 'c', 'c', sysdate, 3000000, 10);

-- Salariul inserat este mai mare decat salariul minim

Oracle SQL Developer : C:\Users\lamer2\OneDrive\Desktop\Aplicati\Facultate\Anul 2\Sisteme de Gestie a Bazelor de Date\Project\Project SGBD\ex11.sql

File Edit View Navigate Run Source Team Tools Window Help

SQL Worksheet History

Reports Worksheet Query Builder

```

58 select * from angajat;
59
60 rollback;
61
62 insert into angajat
63 values (13, 'c', 'c', sysdate, 3000000, 10);
-- Salariul inserat este mai mare decat salariul minim
65
66 alter trigger nivel_linie disable;
67 insert into angajat
68 values (13, 'c', 'c', sysdate, 3000000, 10);
69 alter trigger nivel_linie enable;
70 -- acum orice am inserata, triggerul va arunca o eroare
71
72 insert into angajat

```

Script Output x Query Result x

Error starting at line : 62 in command -
 insert into angajat
 values (13, 'c', 'c', sysdate, 3000000, 10)
 Error report -
 ORA-20001: Salariul inserat este mai mare decat salariul minim
 ORA-06512: at "AMER.NIVEL_LINIE", line 6
 ORA-04088: error during execution of trigger 'AMER.NIVEL_LINIE'

Compiler - Log

Messages Logging Page Statements Compiler

Click on an identifier with the Control key down to perform "Go to Declaration"

Line 62 Column 45 | Insert | Modified: Windows_CP

10:36 AM ENG 22-Dec-21

```
-- Nu este nevoie sa folosim rollback, intrucat ultima inserare a declansat trigger-ul
alter trigger nivel_linie disable;

insert into angajat

values (13, 'c', 'c', sysdate, 3000000, 10);

alter trigger nivel_linie enable;

-- acum orice am inseră, triggerul va arunca o eroare
```

insert into angajat

```
values (14, 'c', 'c', sysdate, 1, 10);
```

-- Salariul inserat este mai mic decat salariul minim

The screenshot shows the Oracle SQL Developer interface with a SQL Worksheet window containing the following code:

```

64 -- Salariul inserat este mai mare decat salariul minim
65
66 alter trigger nivel_linie disable;
67 insert into angajat
68 values (13, 'c', 'c', sysdate, 3000000, 10);
69 alter trigger nivel_linie enable;
70 -- acum orice am inseră, triggerul va arunca o eroare
71
72 insert into angajat
73 values (14, 'c', 'c', sysdate, 1, 10);
74 -- Salariul inserat este mai mic decat salariul minim
75 rollback;
76
77 update angajat
78 set salariu = 300000

```

The line `73 values (14, 'c', 'c', sysdate, 1, 10);` is highlighted in yellow, indicating it is the source of the error.

In the Script Output pane below, the error message is displayed:

```

Error starting at line : 72 in command -
insert into angajat
values (14, 'c', 'c', sysdate, 1, 10)
Error report -
ORA-20001: Salariul inserat este mai mic decat salariul minim
ORA-06512: at "AMER.NIVEL_LINIE", line 8
ORA-04088: error during execution of trigger 'AMER.NIVEL_LINIE'

```

```
rollback;
```

```
update angajat
```

```
set salariu = 300000
```

```
where cod_angajat = 1;
```

```
-- Salariul updatat este mai mare decat salariul mediu
```

The screenshot shows the Oracle SQL Developer interface. In the top-left pane, there is a tree view labeled 'Projects' with several files listed under 'Script'. The main workspace is a 'Worksheet' tab containing the following SQL code:

```
70 -- acum orice am insera, triggerul va arunca o eroare
71
72 insert into angajat
73 values (14, 'c', 'c', sysdate, 1, 10);
74 -- Salariul inserat este mai mic decat salariul minim
75 rollback;
76
77 update angajat
78 set salariu = 300000
79 where cod_angajat = 1;
80 -- Salariul updatat este mai mare decat salariul mediu
81 rollback;
82
83 alter trigger nivel_linie disable;
84 insert into angajat
```

In the bottom-right pane, the 'Script Output' tab displays the execution results and errors:

```
Error starting at line : 77 in command -
update angajat
set salariu = 300000
where cod_angajat = 1
Error report -
ORA-20001: Salariul updatat este mai mare decat salariul mediu
ORA-06512: at "AMER.NIVEL_LINIE", line 12
ORA-04088: error during execution of trigger 'AMER.NIVEL_LINIE'
```

```
rollback;
```

```
alter trigger nivel_linie disable;
```

```
insert into angajat
```

```
values (13, 'c', 'c', sysdate, 3000000, 10);
```

```
alter trigger nivel_linie enable;
```

```
-- acum orice am modifica, triggerul va arunca o eroare
```

```
update angajat
```

```

set salariu = 1
where cod_angajat = 1;
-- Salariul updatat este mai mic decat salariul mediu

```

rollback;

The screenshot shows the Oracle SQL Developer interface with a script editor window containing the following code:

```

82
83 alter trigger nivel_linie disable;
84 insert into angajat
85 values (13, 'c', 'c', sysdate, 3000000, 10);
86 alter trigger nivel_linie enable;
87 -- acum orice am modifica, triggerul va arunca o eroare
88
89 update angajat
90 set salariu = 1;
91 where cod_angajat = 1;
92 -- Salariul updatat este mai mic decat salariul mediu
93
94 rollback;
95
96

```

The line `where cod_angajat = 1;` is highlighted in yellow. Below the code, the "Script Output" tab shows the following error message:

Error starting at line : 89 in command -
update angajat
set salariu = 1
where cod_angajat = 1
Error report -
ORA-20001: Salariul updatat este mai mic decat salariul mediu
ORA-06512: at "AMER.NIVEL_LINIE", line 14
ORA-04088: error during execution of trigger 'AMER.NIVEL_LINIE'

alter trigger nivel_linie disable;

12.

```
/*
```

Definiti un declansator care sa introduca date intr-un tabel creat
dupa ce utilizatorul curent a folosit o comanda LDD
(declansator sistem - la nivel de schema)

```
*/
```

-- Cod Exercitiul 12:

```
create table informatii_tabel(      -- Cream tabelul in care o sa retinem informatiile despre  
utilizator
```

```
    baza_de_date    varchar2(50), -- Numele bazei de date
```

```
    user_logat      varchar2(30), -- Numele utilizatorului
```

```
    eveniment_sistem  varchar2(20), -- Tipul comenzii(CREATE, ALTER, DROP)
```

```
    tip_object      varchar2(30), -- Tipul obiectului asupra caruia a fost aplicata  
instructiunea(TABLE, INDEX)
```

```
    nume_object     varchar2(30), -- Numele obiectului
```

```
    data           timestamp(3)  -- Data la care a fost executata instructiunea/comanda
```

```
);
```

```
create or replace trigger exercitiul_12      -- Cream trigger-ul LDD
```

```
    after create or drop or alter on schema   -- Acesta va actiona dupa comenziile create, alter  
sau drop
```

```
begin
```

```
    insert into informatii_tabel      -- Inseram in tabelul definit anterior
```

```
        values (sys.database_name,      -- Toate valorile generate de sistem
```

```
            sys.login_user,
```

```
            sys.sysevent,
```

```

    sys.dictionary_obj_type,
    sys.dictionary_obj_name,
    systimestamp(3)

);

end;
/

```

-- Rezultate/Explicatii:

```
/*
```

Poza atasata mai jos prezinta doar cateva dintre comenzi pe care le-am rulat dupa ce am trecut de exercitiul 12, comenzi rulate in scopul realizarii/imbunatatirii celorlalte exercitii din cadrul proiectului. In total, sunt peste 600 de astfel de comenzi

```
*/
```

BAZA_DE_DATE	USER_LOGAT	EVENIMENT_SYSTEM	TIP_OBJECT	NUME_OBJECT	DATA
631	O1LG	AMER	DROP	TABLE	TARA
632	O1LG	AMER	DROP	TABLE	PARTICIPA
633	O1LG	AMER	DROP	TABLE	STUDIAZA
634	O1LG	AMER	DROP	TABLE	PROGRAMA
635	O1LG	AMER	CREATE	INDEX	PK_TARA
636	O1LG	AMER	CREATE	TABLE	TARA
637	O1LG	AMER	CREATE	INDEX	PK_LOCATIE
638	O1LG	AMER	CREATE	TABLE	LOCATIE
639	O1LG	AMER	CREATE	INDEX	PK_FACULTATE
640	O1LG	AMER	CREATE	TABLE	FACULTATE
641	O1LG	AMER	CREATE	INDEX	PK_SPECIAL...
642	O1LG	AMER	CREATE	TABLE	SPECIALIZARE
643	O1LG	AMER	CREATE	INDEX	PK_CAMIN
644	O1LG	AMER	CREATE	TABLE	CAMIN
645	O1LG	AMER	CREATE	INDEX	PK_STUDENT
646	O1LG	AMER	CREATE	TABLE	STUDENT
647	O1LG	AMER	CREATE	INDEX	PK_STUDIAZA
648	O1LG	AMER	CREATE	TABLE	STUDIAZA
649	O1LG	AMER	CREATE	INDEX	PK_DISCIPLINA
650	O1LG	AMER	CREATE	TABLE	DISCIPLINA
651	O1LG	AMER	CREATE	INDEX	PK_EXMINARE
652	O1LG	AMER	CREATE	TABLE	EXMINARE
653	O1LG	AMER	CREATE	INDEX	PK_PROGRAMA
654	O1LG	AMER	CREATE	TABLE	PROGRAMA
655	O1LG	AMER	CREATE	INDEX	PK_ANGAJAT
656	O1LG	AMER	CREATE	TABLE	ANGAJAT
657	O1LG	AMER	ALTER	TABLE	ANGAJAT
658	O1LG	AMER	CREATE	INDEX	PK_SECRETARA
659	O1LG	AMER	CREATE	TABLE	SECRETARA
660	O1LG	AMER	CREATE	INDEX	PK_PROFESOR
661	O1LG	AMER	CREATE	TABLE	PROFESOR
662	O1LG	AMER	CREATE	INDEX	PK_PARTICIPA
663	O1LG	AMER	CREATE	TABLE	PARTICIPA

```
/*
```

Exemple de comenzi rulate imediat dupa crearea tabelului informatii_tabel:

```
*/
```

```
create table exemplu_tabel (
    col_1 number(2)
);
```

```
alter table exemplu_tabel
add (col_2 number(2));
```

```
insert into exemplu_tabel
values (15, 32);
```

```
create index ind_exemplu_tabel
on exemplu_tabel(col_1);
```

The screenshot shows the Oracle SQL Developer interface with a script editor containing the following SQL code:

```
22 create table exemplu_tabel (
23     col_1 number(2)
24 );
25
26 alter table exemplu_tabel
27 add (col_2 number(2));
28
29 insert into exemplu_tabel
30 values (15, 32);
31
32 create index ind_exemplu_tabel
33 on exemplu_tabel(col_1);
34
35 select * from informatii_tabel;
```

The script is executed, and the results are displayed in the Results tab:

```
Index IND_EXEMPLU_TABEL created.
```

BAZA_DE_DATE	USER_LOGAT	EVENIMENT_SISTEM	TIP_OBIECT	NUME_OBJECT
O11G	AMER	CREATE	TABLE	EXEMPLU_TABEL
O11G	AMER	ALTER	TABLE	EXEMPLU_TABEL
O11G	AMER	CREATE	INDEX	IND_EXEMPLU_T

The screenshot shows the Oracle SQL Developer interface. In the top-left corner, there's a message bar: "Oracle SQL Developer : C:\Users\lamer2\OneDrive\Desktop\Aplicatiile Facultate\Anul 2\Sisteme de Gestie a Bazelor de Date\Project\Project SGBD\ex12.sql". The main window has a toolbar at the top with various icons. Below the toolbar is a menu bar: File, Edit, View, Navigate, Run, Source, Team, Tools, Window, Help. A tab bar below the menu shows several open files: ex7.sql, ex8.sql, ex9.sql, Welcome Page, ex5.sql, ex10.sql, ex11.sql, ex12.sql, and ex13.sql. The central area is a "Worksheet - Query Builder" pane where the following SQL code is displayed:

```

22 create table exemplu_tabel (
23     col_1 number(2)
24 );
25
26 alter table exemplu_tabel
27 add (col_2 number(2));
28
29 insert into exemplu_tabel
30 values (15, 32);
31
32 create index ind_exemplu_tabel
33 on exemplu_tabel(col_1);
34
35 select * from informatii_table;

```

Below the code, a "Script Output" pane shows the results of the execution:

USER_LOGAT	EVENIMENT_SISTEM	TIP_OBJECT	NUME_OBJECT	DATA
AMER	CREATE	TABLE	EXEMPLU_TABEL	22-DEC-21 12.05.36.432000000 PM
AMER	ALTER	TABLE	EXEMPLU_TABEL	22-DEC-21 12.05.36.491000000 PM
AMER	CREATE	INDEX	IND_EXEMPLU_TABEL	22-DEC-21 12.05.36.524000000 PM

At the bottom of the interface, there's a status bar with the text "Compiler - Log" and "Click on an identifier with the Control key down to perform "Go to Declaration"".

/*

Observam ca doar comenzile de create si alter au fost inserate in tabel, nu si comanda de insert.

*/

13.

-- Pachet ce contine subprogramele exercitiilor 6-9.

-- Cod Exercitiul 13:

```
create or replace package exercitiul_13
```

```
is
```

```
procedure exercitiul_6(id    student.cod_student%type);
```

```
procedure exercitiul_7(a  angajat.salariu%type, b  angajat.salariu%type);
```

```
function exercitiul_8(
```

```
    nume_student      in  student.nume%type,
```

```
    prenume_student   in  student.prenume%type
```

```
)
```

```
    return tara.nume%type;
```

```
procedure exercitiul_9(ranking_facultate  facultate.ranking%type);
```

```
end exercitiul_13;
```

```
/
```

```
create or replace package body exercitiul_13
```

```
is
```

```
procedure exercitiul_6(
    id      student.cod_student%type      -- codul studentului dat ca parametru
)
is
    cursor c is select rownum, cod_student, nume, prenume from student; -- cursor pentru
    a parcurge liniile din tabelul student
```

```
    nr_linie_id      number;           -- numarul liniei parcuse de cursor
    id_student       student.cod_student%type; -- id-ul studentului parcurs de
cursor
    nume            student.nume%type; -- numele studentului parcurs de
cursor
    prenume          student.prenume%type; -- prenumele studentului
parcurs de cursor
```

```
    id_maxim        student.cod_student%type; -- id-ul de valoare maxima a studentilor
aflati pe liniile urmatoare id-ului de student dat ca parametru
```

```
type tab_ind_fibonacci is table of number index by pls_integer; -- tablou indexat ce
memoreaza numerele fibonacci mai mici +1 decat id_maxim
```

```
    ind      tab_ind_fibonacci; -- tabloul indexat in care o sa memoram numerele
fibonacci calculate
```

```
    contor      number;           -- contor folosit pentru a parurge tabloul indexat
anterior declarat
    nr_studenti   number;           -- numarul total de studenti
    sem          boolean;          -- semafor folosit pentru eficientizarea timpului de
executie
```

```

no_fibonacci_found exception;          -- exceptie specifica exercitiului
too_many_fibonacci exception;         -- exceptie specifica exercitiului

type detalii_profesor is record(      -- tip inregistrare(record) pentru memorarea
detaliilor despre un profesor
    nume    angajat.nume%type,
    prenume  angajat.prenume%type,
    salariu angajat.salariu%type
);

type v_detalii_profesor is varray(25) of detalii_profesor;   -- varray de 25 de
detaliii_profesor

type tab_imb_nume_profesori is table of v_detalii_profesor; -- tablou imbricat al carui
elemente sunt varray-uri de detalii_profesor(matrice)

imb tab_imb_nume_profesori := tab_imb_nume_profesori();        -- imb este practic o
matrice de detalii_profesor

begin
    -- verificam daca id-ul dat ca parametru exista de fapt in baza de date
    select cod_student into id_student from student where cod_student = id;
    -- generam NO_DATA_FOUND daca nu gasim id-ul dat ca parametru
    -- Daca nu s-a generat eroarea NO_DATA_FOUND inseamna ca id-ul exista si il avem
memorat in variabila auxiliara id_student

open c;  -- deschidem cursorul pentru a incepe parcurgerea tabelului student

```

```

loop -- cat timp nu am parcurs tot tabelul

    fetch c into nr_linie_id, id_student, nume, prenume; -- plasam valorile liniei curente
    in variabilele asociate

        exit when c%notfound; -- oprim bucla atunci cand am trecut de
        ultima linie a tabelului

        if id_student = id then -- daca id-ul curent este egal cu id-ul dat ca
        parametru

            exit; -- inseamna ca putem sa iesim din bucla parcurgerii
            cursorului

            end if;

        end loop;

        -- am gasit linia unde se afla student-ul cu id-ul dat
        -- trebuie sa calculam id-ul maxim dintre cei ramasi

        id_maxim := id_student; -- initial, id-ul maxim va fi fix id-ul dat ca parametru

    loop -- incepand cu linia imediat urmatoare liniei in care se afla id-
    ul dat ca parametru

        exit when c%notfound; -- verificam intai daca id-ul dat ca parametru se afla
        deja pe ultima linie a tabelului student

        fetch c into nr_linie_id, id_student, nume, prenume; -- Daca nu era pe ultima linie,
        atunci incepem cautarea id-ului maxim

        if id_student > id_maxim then -- Daca id-ul curent este mai mare decat id-ul
        maxim calculat pana in acest moment

            id_maxim := id_student; -- Atunci noul id_maxim devine id-ul curent

            end if;

    end loop;

```

```

dbms_output.put_line('ID-ul dat: ' || id);      -- Printam id-ul dat ca parametru

dbms_output.put_line('ID-ul maxim: ' || id_maxim); -- Printam id-ul maxim incepand cu
linia urmatoare id-ului curent

close c;

dbms_output.new_line;      -- Printam o noua linie

-- Acum ca avem id-ul maxim calculat, putem incepe calcularea numerelor fibonacci mai
mici +-1 decat id_maxim

if 1 <= id_maxim + 1 then    -- Daca primul termen fibonacci(1) este mai mic sau egal decat
id_maxim + 1 atunci putem incepe calcularea

    ind(1) := 1;            -- Primul termen fibonacci este 1
    ind(2) := 1;            -- Al doilea termen fibonacci este 2
    contor := 3;            -- Calculul incepe cu al treilea termen fibonacci, deci contor este 3
    else                    -- Daca in schimb id_maxim + 1 este mai mic decat primul termen fibonacci,
    atunci nu avem ce termeni fibonacci sa calculam(numerele fibonacci sunt strict pozitive)

        raise no_fibonacci_found;    -- Prin urmare, aruncam exceptia no_fibonacci_found,
intrucat nu avem numere fibonacci

    end if;

-- Daca nu s-a aruncat exceptia, atunci putem continua calcularea numerelor fibonacci

    while ind(contor-1) + 1 <= id_maxim loop          -- Cat timp termenul anterior + 1 este
mai mic sau egal decat id_maxim calculam urmatorul termen fibonacci

        if ind(contor - 1) + ind(contor - 2) > id_maxim then  -- Daca urmatorul termen fibonacci
va fi mai mare strict decat id_maxim, nu il mai calculam

            exit;                -- Ci in schimb iesim din bucla

```

```

end if;

ind(contor) := ind(contor-1) + ind(contor-2); -- fib(i) = fib(i-1) + fib(i-2);

contor := contor + 1; -- i++;

end loop;

dbms_output.put_line('Numerele fibonacci mai mici ca ' || id_maxim || ':'); -- Incepem
afisarea termenilor fibonacci calculati

for i in ind.first..ind.last loop -- Parcurgem tabloul indexat folosind o bucla for
    dbms_output.put(ind(i) || ' '); -- Printam toti termenii fibonacci intr-o singura linie
separand prin spatiu

end loop;

dbms_output.new_line; -- Afisam 3 linii noi pentru a separa continutul afisat
dbms_output.new_line;
dbms_output.new_line;

-- Determinam numarul de studenti care se afla dupa id-ul dat, precum si numarul de
studenti al caror id este fibonacci

nr_studenti := 0; -- Initial consideram ca numarul de studenti aflati sub id-ul dat
este 0

contor := 0; -- In contor vom memora numarul de id-uri fibonacci, contor este
initializat cu 0

open c; -- Deschidem cursorul din nou, acesta se afla inainte de prima linie din
tabel

loop -- Cautam sa ajungem la linia unde se afla id-ul dat ca parametru
    fetch c into nr_linie_id, id_student, nume, prenume;

    exit when id_student = id; -- Odata ce am ajuns la linia cu id-ul dat ca parametru
oprim bucla

```

```

end loop;

loop          -- Incepem sa numaram studentii aflati sub id-ul dat

    nr_studenti := nr_studenti + 1; -- Luam in considerare si id-ul dat ca fiind unul elegibil

    -- Testam daca id-ul curent este fibonacci

    sem := true;           -- Consideram sem = true atunci cand id-ul nu este fibonacci

    for i in ind.first..ind.last loop    -- Parcurgem tabloul de numere fibonacci

        if id_student >= ind(i) - 1 and id_student <= ind(i) + 1 then    -- Daca id-ul curent
            este egal +-1 fata de termenul fibonacci curent

                contor := contor + 1;    -- Incrementam contor-ul pentru a creste numarul de id-
            uri fibonacci

                sem := false;         -- sem devine false, intrucat stim ca id-ul este fibonacci

            end if;

        if sem = false then      -- Daca am determinat ca id-ul este fibonacci, nu mai este
            nevoie sa comparam cu restul numerelor fibonacci din tablou

            exit;                 -- Prin urmare, putem iesi din bucla for

        end if;

    end loop;

    fetch c into nr_linie_id, id_student, nume, prenume;    -- Trecem la urmatorul id

    exit when c%notfound;           -- Repetam procesul pentru fiecare id in
parte

    end loop;

    close c;   -- Inchidem cursorul odata ce am terminat

```

```
/*
```

Pana acum avem urmatoarele:

```
contor = numarul de studenti cu id fibonacci
```

```
nr_studenti = numarul tuturor studentilor
```

```
*/
```

```
-- dbms_output.put_line(contor);
```

```
-- dbms_output.put_line(nr_studenti);
```

```
if contor = 0 then           -- daca contor = 0 inseamna ca nu avem id-uri fibonacci si  
aruncam exceptia no_fibonacci_found
```

```
    raise no_fibonacci_found;
```

```
elsif contor > 0.5 * nr_studenti then   -- daca avem mai mult de jumata + 1 dintre toti  
studentii eligibil cu id-uri fibonacci, intram pe exceptia TOO_MANY_FIBONACCI
```

```
    raise too_many_fibonacci;
```

```
end if;
```

```
-- daca nu am intrat pe exceptia TOO_MANY_FIBONACCI, inseamna putem sa
```

```
-- afisam toti studentii care se incadreaza in conditia id-ului:
```

```
-- Practic nu mai avem nevoie de numarul de id-uri fibonacci, deci putem sa reutilizam  
variabila contor
```

```
open c;  -- Deschidem cursorul pentru parcurgerea tabelului student
```

```
loop
```

```
    fetch c into nr_linie_id, id_student, nume, prenume;
```

```
    exit when id_student = id;      -- Parcurgem toate liniile pana la linia cu id-ul dat ca  
parametru
```

```
end loop;
```

```
-- Incepem sa construim matricea de detalii_profesor din imb
```

```
contor := 1; -- Contor ia primul indice al tabloului imb
```

```
loop
```

```
-- Pentru fiecare id de dupa id-ul parametru, verificam daca este fibonacci si afisam datele
```

```
for i in ind.first..ind.last loop
```

```
if id_student >= ind(i) - 1 and id_student <= ind(i) + 1 then -- Daca id-ul este id fibonacci afisam informatiile
```

```
    dbms_output.put_line('Nr linie: ' || nr_linie_id);
```

```
    dbms_output.put_line('ID: ' || id_student);
```

```
    dbms_output.put_line('Nume Student: ' || nume || ' ' || prenume);
```

```
    dbms_output.put('Profesori: ');
```

```
imb.extend; -- necesar tablourilor imbicate si varray-urilor
```

```
select a.nume, a.prenume, a.salariu -- Selectam toti profesorii studentului cu id-ul curent
```

```
bulk collect into imb(contor) -- Stocam toate campurile selectate in imb(contor), care este de tipul varray(25) de detalii_profesor
```

```
from participa p join student s on (s.cod_student = p.cod_student)
```

```
join profesor pf on (p.cod_angajat = pf.cod_angajat)
```

```
join angajat a on (p.cod_angajat = a.cod_angajat)
```

```
where s.cod_student = id_student
```

```
order by s.nume;
```

```
for i in imb(contor).first..imb(contor).last loop      -- Parcurgem lista tuturor
profesorilor unui student

    dbms_output.put(imb(contor)(i).nume || ' ' || imb(contor)(i).prenume || ' ' ||
imb(contor)(i).salariu); -- Afisam numele, prenumele si salariul profesorului curent

    if i < imb(contor).last then      -- Daca nu am ajuns la ultimul profesor

        dbms_output.put(',');      -- Afisam virgula ca separator

    else

        dbms_output.put('.');      -- Daca am ajuns la ultimul profesor, punem
simbolul punct.

    end if;

end loop;
```

```
contor := contor + 1;      -- Incrementam contor-ul pentru a trece la urmatorul
element al lui imb
```

```
dbms_output.new_line;      -- Punem new_line pentru a se afisa toate
dbms_output.put()-urile executate in for

dbms_output.new_line;      -- Punem inca un new_line pentru a pastra afisarea
mai spatiata
```

```
end if;
```

```
end loop;
```

```
fetch c into nr_linie_id, id_student, nume, prenume;      -- Trecem la urmatorul
student
```

```
exit when c%notfound;          -- Incheiem bucla atunci cand am
terminat studentii
```

```
end loop;
```

```
close c;
```

```
ind.delete; -- stergem tabloul indexat
```

```
for i in imb.first..imb.last loop
```

```
    imb(i).delete; -- Stergem fiecare varray existent in imb
```

```
end loop;
```

```
imb.delete; -- Stergem tabloul imbricat
```

```
exception
```

```
    when no_data_found then dbms_output.put_line('ID-ul dat nu exista in baza de date');
```

```
-- Atunci cand avem no_data_found inseamna ca nu exista student cu id-ul dat ca parametru
```

```
    when no_fibonacci_found then dbms_output.put_line('Niciun student nu are id fibonacci');
```

```
-- Atunci cand nu exista niciun id fibonacci sau daca id_maxim este mai mic decat primul termen  
fibonacci
```

```
    when too_many_fibonacci then dbms_output.put_line('Mai mult de jumata + 1 dintre  
studentii eligibili au id-uri fibonacci'); -- Atunci cand mai mult de jumata + 1 dintre id-uri sunt  
fibonacci primim aceasta eroare
```

```
        -- nu intalnim erori de tipul too_many_rows, intrucat toate selecturile pe care le
```

```
        -- efectuam se fac impreuna cu clauza where cod_student = id;
```

```
        -- din moment ce prin definitie cod_student este cheie primara in tabela student
```

```
        -- o sa avem cel mult un rezultat in urma select-ului.
```

```
        -- when too_many_rows then dbms_output.put_line('Too Many Rows');
```

```
when others then dbms_output.put_line('Alt tip de eroare!');      -- Tratam orice alt tip de
eroare  
  
end exercitiul_6;
```

```
procedure exercitiul_7(  
    a_angajat.salariu%type,      -- Limita inferioara a intervalului salarial  
    b_angajat.salariu%type      -- Limita superioara a intervalului salarial  
)  
is  
    -- cursor explicit parametrizat  
    -- selecteaza codul, numele, prenumele, salariul profesorilor ca se incadreaza in intervalul  
    -- salarial  
    -- precum si denumirea facultatii la care acesti profesori predau  
    cursor c(x_angajat.salariu%type, y_angajat.salariu%type) is  
        select cod_angajat, nume, prenume, salariu, denumire  
        from angajat join facultate using(cod_facultate)  
        where salariu between x and y;  
  
    a_cod      angajat.cod_angajat%type;  -- variabila pentru codul profesorului  
    a_nume     angajat.nume%type;       -- variabila pentru numele profesorului  
    a_prenume   angajat.prenume%type;   -- variabila pentru prenumele profesorului  
    a_salariu   angajat.salariu%type;  -- variabila pentru salariul profesorului
```

```
f_denumire      facultate.denumire%type; -- variabila pentru denumirea facultatii la care
preda profesorul
```

```
nr_linii      number := 0; -- numarul de iteratii ale cursorului c
salariu_inexistent exception; -- exceptie pentru cazul in care nu exista salariu din
intervalul [a, b]
```

```
begin
```

```
open c(a, b); -- Deschidem cursorul c pentru parametri a si b
```

```
loop
```

```
    fetch c into a_cod, a_nume, a_prenume, a_salariu, f_denumire; -- Asignam
    valorile linie curente variabilelor corespunzatoare
```

```
    exit when c%notfound; -- Folosim atributul notfound pentru a verifica daca am
    parcurs toate liniile tabelului. In caz afirmativ iesim din bucla
```

```
        nr_linii := nr_linii + 1; -- La fiecare iteratie a cursorului, incrementam variabila
        nr_linii
```

```
    end loop;
```

```
close c; -- Dupa ce am iterat prin toate liniile tabelului inchidem cursorul
```

```
if nr_linii = 0 then -- Daca cursorul c nu a gasit nicio linie din tabel, inseamna ca nu
exista un salariu din intervalul [a, b]
```

```
    raise salariu_inexistent; -- Prin urmare, se arunca exceptia "salariu_inexistent"
```

```
end if;
```

```
-- Daca nu s-a aruncat exceptia "salariu_inexistent"
```

```
open c(a, b); -- Putem incepe afisarea tuturor angajatilor cu salariul din intervalul
[a, b]
```

```
loop
```

```
    fetch c into a_cod, a_nume, a_prenume, a_salariu, f_denumire; -- Extragem  
informatiile angajatului curent
```

```
    exit when c%notfound; -- atribut notfound
```

```
    dbms_output.put_line('Nume Angajat: ' || a_nume || ' ' || a_prenume); -- Afisam  
informatiile angajatului
```

```
    dbms_output.put_line('Salariu: ' || a_salariu);
```

```
    dbms_output.put_line('Facultate: ' || f_denumire);
```

```
    dbms_output.put('Colegi: ');
```

```
-- Obtinem lista colegilor angajatului curent folosindu-ne de un ciclu cursor in bucla for
```

```
for i in (
```

```
    select nume, prenume
```

```
    from angajat join facultate using(cod_facultate) -- ciclu cursor
```

```
    where denumire = f_denumire and cod_angajat <> a_cod
```

```
) loop
```

```
    dbms_output.put(i.nume || ' ' || i.prenume || ','); -- Afisam lista tuturor  
colegilor angajatului curent
```

```
end loop;
```

```
dbms_output.new_line; -- Afisam new_line pentru a se afisa lista anterior mentionata
```

```
dbms_output.new_line; -- Afisam inca 2 new_line-uri pentru spatiere
```

```
dbms_output.new_line;
```

```
end loop;
```

```
close c; -- Inchidem cursorul c

exception

  -- nu intalnim cazul in care sa avem eroarea no_data_found

  -- intrucat in cazul cursoarelor, chiar daca nu se gaseste nicio linie in urma

  -- select-urilor, cursoarele raman valide, dar goale

  -- in cazul care cursoarelor goale, avem exceptia "salariu_inexistent"

  -- when no_data_found then dbms_output.put_line('Nu au fost gasite date in baza de
date');

  -- nu intalnim cazul in care sa avem eroarea too_many_rows

  -- intrucat toate selecturile facute sunt pentru cursoare

  -- when too_many_rows then dbms_output.put_line('Too Many Rows');

when salariu_inexistent then dbms_output.put_line('Nu exista niciun salariu din intervalul
[' || a || ',' || b || ']');

when others then dbms_output.put_line('Alt tip de eroare!');

end exercitiul_7;
```

```

function exercitiul_8(
    nume_student      in student.nume%type,          -- Numele studentului dat ca
parametru de intrare de tip IN
    prenume_student   in student.prenume%type       -- Prenumele studentului dat ca
parametru de intrare de tip IN
)
return tara.nume%type      -- Tipul de data returnat este numele tarii in care se afla
facultatea de ranking maxim a studentului dat ca parametru
is
    cod           student.cod_student%type;        -- Codul studentului dat ca parametru
    nume_tara     tara.nume%type;                  -- Numele tarii in care afla facultatea
studentului
    nume_specializare specializare.denumire%type;  -- Numele specializarii la care este
inscris studentul
    nume_camin    camin.denumire%type;            -- Numele caminului studentului
begin
    -- putem avea erorile no_data_found sau too_many_rows
    -- no_data_found atunci cand nu exista un student cu numele si prenumele dat
    -- too_many_rows atunci cand avem cel putin doi studenti cu nume si prenume identice
select cod_student    -- selectam codul studentului cu numele si prenumele date
into cod
from student
    where initcap(nume) = initcap(nume_student) and initcap(prenume) =
initcap(prenume_student);
    -- Selectarea se face fara case sensitivity, numele studentului putand fii scris cu numere
mici sau mari.

    -- Daca nu s-a aruncat nicio exceptie, atunci putem cauta numele caminului, numele
specializarii si numele tarii
    -- Din moment ce selectam informatii din minim 3 tabele diferite,

```

-- avem de facut join-uri care sa conecteze informatiile astfel incat
-- sa se pastreze corectitudinea datelor.

```
select c.denumire "Camin", sp.denumire "Specializare", t.nume "Tara"  
into nume_camin, nume_specializare, nume_tara  
from student s  
join camin c on (s.cod_camin = c.cod_camin)  
join specializare sp on (s.cod_specializare = sp.cod_specializare)  
join studiaza st on (s.cod_student = st.cod_student)  
join facultate f on (f.cod_facultate = st.cod_facultate)  
join locatie l on (l.cod_locatie = f.cod_locatie)  
join tara t on (l.cod_tara = t.cod_tara)  
where f.rank = (  
    select min(ranking) -- cu cat rankingul este mai mic cu atat facultatea este mai de top  
    from student s2  
    join studiaza st2 on (s2.cod_student = st2.cod_student)  
    join facultate f2 on (f2.cod_facultate = st2.cod_facultate)  
    where s2.cod_student = cod -- Selectam ranking-ul maxim dintre toate facultatiile  
    urmate de studentul dat ca parametru  
    ) and s.cod_student = cod; -- Selectam detaliile despre facultatea al carei ranking este egal  
    cu ranking-ul maxim dintre toate facultatile urmate de studentul dat ca parametru  
  
    -- Afisam detaliile obtinute  
    dbms_output.put('Studentul ' || nume_student || ' ' || prenume_student);  
    dbms_output.put(' urmeaza specializarea ' || nume_specializare);  
    dbms_output.put(' dintr-o facultate din ' || nume_tara || ',');  
    if nume_camin = null then -- Daca studentul nu este cazat la vreun camin, afisam un mesaj  
        sugestiv
```

```

dbms_output.put(' studentul nefiind cazat la vreun camin.');
else
    dbms_output.put(' fiind cazat in caminul ' || nume_camin || '.');
end if;

dbms_output.new_line; -- Pentru dbms_output.put();

return(nume_tara); -- Functia returneaza numele tarii in care se afla facultatea de
ranking maxim urmata de student

exception
when no_data_found then      -- Atunci cand nu exista studentul dat ca parametru in baza
de date
    dbms_output.put_line('Nu exista niciun student cu numele de "' || nume_student || ''
|| prenume_student || "'");
    return("");
-- Returnam sirul vid, intrucat nu putem afla tara in care se afla
facultatea
when too_many_rows then      -- Exista mai multi studenti cu acelasi nume si prenume
    dbms_output.put_line('Exista mai multi studenti cu numele de "' || nume_student || ''
|| prenume_student || "'");
    return("");
when others then           -- Orice alt tip de eroare
    dbms_output.put_line('Alt tip de eroare!');
    return("");
end exercitiul_8;

```

```
procedure exercitiul_9(
    ranking_facultate facultate.ranking%type      -- ranking-ul facultatii dat ca parametru
)
is
    cod          facultate.cod_facultate%type;    -- Codul facultatii
    nume_facultate   facultate.denumire%type;      -- Numele facultatii
    nota_maxima     examinare.nota%type;           -- Nota maxima obtinuta in cadrul
facultatii
    nume_disciplina  disciplina.denumire%type;    -- Numele disciplinei asupra carei s-a
obtinut nota maxima
    nume_student     student.nume%type;           -- Numele studentului care a obtinut
nota maxima la disciplina din cadrul facultatii
    nume_profesor    angajat.nume%type;           -- Numele profesorului coordonator
disciplinei cu nota maxima
```

```
begin
```

```
-- verificam daca codul exista
```

```
select cod_facultate
```

```
into cod
```

```
from facultate
```

```
where ranking = ranking_facultate;
```

```
-- Daca codul nu ar fi exista, s-ar fi generat eroare no_data_found si s-ar fi oprit executia
procedurii
```

```
select *  -- Selectam informatiile necesare din tabele multiple folosind join-uri
```

```
into nume_facultate, nota_maxima, nume_disciplina, nume_student, nume_profesor
```

```
from(
```

```

select f.denumire "Facultate",
      e.nota "Nota",
      d.denumire "Disciplina",
      s.nume || ' ' || s.prenume "Student", -- Numele studentului si profesorului sunt
complete
      a.nume || ' ' || a.prenume "Profesor"

from student s

join participa p on (p.cod_student = s.cod_student)

join examinare e on (e.cod_disciplina = p.cod_disciplina)

join angajat a on (a.cod_angajat = p.cod_angajat)

join profesor pf on (a.cod_angajat = pf.cod_angajat)

join disciplina d on (p.cod_disciplina = d.cod_disciplina)

join programa pr on (d.cod_disciplina = pr.cod_disciplina)

join specializare sp on (pr.cod_specializare = sp.cod_specializare)

join facultate f on (f.cod_facultate = sp.cod_facultate)

where nota = ( -- Selectam toate detaliile pentru disciplina cu nota maxima

select max(nota) -- Subcerere in care electam nota maxima din cadrul facultatii date

from examinare e

join disciplina d on (e.cod_disciplina = d.cod_disciplina)

join programa p on (d.cod_disciplina = p.cod_disciplina)

join specializare sp on (sp.cod_specializare = p.cod_specializare)

join facultate f on (sp.cod_facultate = f.cod_facultate)

where f.cod_facultate = cod

) and f.cod_facultate = cod

order by s.nume, s.prenume -- Ordonam selectia dupa nume si prenume

)

where rownum = 1; -- In cazul in care exista mai multi studenti, selectam doar primul
ordonat dupa nume si prenume

```

```
-- Afisam informatiile obtinute
```

```
dbms_output.put_line('Facultate: ' || nume_facultate);
dbms_output.put_line('Nota: ' || nota_maxima);
dbms_output.put_line('Disciplina: ' || nume_disciplina);
dbms_output.put_line('Student: ' || nume_student);
dbms_output.put_line('Profesor: ' || nume_profesor);
```

```
exception
```

```
when no_data_found then dbms_output.put_line('Nu exista facultate cu ranking-ul dat');
when too_many_rows then dbms_output.put_line('Exista mai multe facultati care au
ranking-ul egal cu ' || ranking_facultate);
```

```
when others then dbms_output.put_line('Alt tip de eroare!');
```

```
end exercitiul_9;
```

```
end exercitiul_13;
```

```
/
```

-- Rezultate/Explicatii:

```
declare
    a tara.nume%type;
begin
    exercitiul_13.exercitiul_6(11);
    exercitiul_13.exercitiul_7(3800, 5000);

    a := exercitiul_13.exercitiul_8('Mircea', 'Bravo');
    dbms_output.put_line(a);
    exercitiul_13.exercitiul_9(1);
end;
```

/

```
declare
    a tara.nume%type;
begin
    exercitiul_13.exercitiul_6(11);
    exercitiul_13.exercitiul_7(3800, 5000);

    a := exercitiul_13.exercitiul_8('Mircea', 'Bravo');
    dbms_output.put_line(a);
    exercitiul_13.exercitiul_9(1);
end;
```

Output from the DBMS Output window:

- Numerele fibonacci mai mici ca 55:
1 1 2 3 5 8 13 21 34 55
- Mai mult de jumataate + 1 dintre studentii eligibili au id-uri fibonacci
Nume Angajat: Mihaleascu Andrei
Salariu: 3800
Facultate: Facultatea de Matematica si Informatica
Colegi: Popescu Ion, Stefanescu Teodora,
- Nume Angajat: Marinescu Elena
Salariu: 4100
Facultate: Facultatea de Agronomie
Colegi: Popa Andra, Mihai George,
- Nume Angajat: Bogdan Octavian
Salariu: 5000
Facultate: Facultatea de Medicina
Colegi: Alexandrescu Marius,
- Nu exista niciun student cu numele de "Mircea Bravo"
- Facultate: Facultatea de Matematica si Informatica
Nota: 10
Disciplina: Matematica
Student: Georgescu Nicolae

/*

Exemplile rulate in cadrul blocului anonim PL/SQL sunt exemplele execute in cadrul exercitiilor 6-9.

*/

14.

/*

Sa se selecteze tara in care se afla facultatea studentului al carui salariu profesoral cumulativ este maximal.

Definim salariul profesoral cumulativ al unui student ca fiind suma tututor salariilor profesorilor la care studentul invata.

Salariu profesoral cumulativ maximal reprezinta multimea profesorilor al caror salarii combinate este cea mai mare.

Daca exista mai multi studenti ai caror salarii cumulative sunt egale se va lua in cosiderare primul in ordinea alfabetica dupa nume si prenume.

Dupa ce se afiseaza tara, sa se afiseze nota maxima obtinuta in cadrul facultatii precum si numele materiei, numele profesorului coordonator, numele studentului care a obtinut nota respectiva si, daca exista, caminul la care acesta este cazat.

Daca sunt mai multe facultati la care este inscris studentul, se va lua in considerare doar prima facultate la care s-a inscris.

pasi:

Pentru fiecare student, printam lista profesorilor si salariilor acestora

Pentru fiecare student, calculam salariul profesoral cumulativ

Selectam studentul cu salariul profesoral cumulativ maximal

*/

-- Cod Exercitiul 14:

create or replace package exercitiul_14

is

cursor studenti is select * from student; -- Cursor pentru parcurgerea tabelului student

-- Cursor parametrizat pentru parcurgerea listei tuturor profesorilor unui student cu codul dat ca parametru

-- Acest cursor va fi folosit in afiseaza_profesori()

```
cursor detalii_profesor(c_student student.cod_student%type)is
```

```
    select a.nume, a.prenume, a.salariu
```

```
        from angajat a join profesor prof on (a.cod_angajat = prof.cod_angajat)
```

```
            join participa p on (a.cod_angajat = p.cod_angajat)
```

```
        where p.cod_student = c_student
```

```
        order by a.nume, a.prenume, a.salariu desc;
```

-- Tip inregistrare(record) folosit pentru extragerea informatiilor(fetch) din cursorul detalii_profesor

-- Aceasta va fi folosit in afiseaza_profesori()

```
type rec_detalii_profesor is record(
```

```
    nume    angajat.nume%type,
```

```
    prenume angajat.prenume%type,
```

```
    salariu angajat.salariu%type
```

```
);
```

-- Tablou indexat de inregistrari de tip rec_detalii_profesor

-- Aceasta va fi folosit in afiseaza_profesori()

```
type ind_profesori is table of rec_detalii_profesor index by pls_integer;
```

ind ind_profesori; -- Declaram un tablou indexat de tipul anterior mentionat, folosit in afiseaza_profesori()

-- Tablou indexat de salarii cumulative, folosit in afiseaza_profesori()

```
type salarii_cumulative is table of angajat.salariu%type index by pls_integer;
```

```
ind_salarii salarii_cumulative; -- Declaram un tablou indexat de tipul anterior mentionat,  
folosit in afiseaza_profesori()
```

```
type salariu_maxim is record(           -- Tip de data complex ce memoreaza detaliile  
angajatului ce are salariul maxim
```

```
cod     student.cod_student%type,  
nume    student.nume%type,  
prenume student.prenume%type,  
salariu angajat.salariu%type  
);
```

```
maxim salariu_maxim; -- Declaram un obiect de tipul record anterior mentionat
```

```
procedure afiseaza_detalii_studenti; -- Procedura ce afiseaza detaliile despre studenti si  
apeleaza afiseaza_profesori()
```

```
function afiseaza_profesori(c_student in student.cod_student%type) return boolean; --  
Functie afiseaza detaliile despre toti profesorii la care invata studentul cu codul dat ca  
parametru, returneaza true daca studentul are cel putin un profesor, false in caz contrar
```

```
function salariu_cumulativ_maxim return student.cod_student%type; -- Functie ce  
afiseaza detaliile despre salariul cumulativ maximal determinat in prin intermediul apelarii  
succesive a functiei afiseaza_profesori() in cadrul procedurii afiseaza_detalii_studenti()
```

```
function tara_facultate(c_student in student.cod_student%type) return  
facultate.cod_facultate%type; -- Functie ce afiseaza tara primei facultati la care este inscris  
studentul al carui cod este dat ca parametru si returneaza codul respectivei facultati
```

```
procedure nota_maxima(c_facultate facultate.cod_facultate%type); -- Procedura ce afiseaza nota maxima obtinuta in cadrul facultatii al carei cod este data ca parametru, impreuna cu restul detaliilor legate de student si profesorul coordonator
```

```
function main return boolean; -- Functia principala din care se vor apela toate subprogramele in cascada
```

```
end exercitiul_14;
```

```
/
```

```
create or replace package body exercitiul_14
```

```
is
```

```
procedure afiseaza_detalii_studenti
```

```
is
```

```
    contor number := 1; -- Contor folosit strict pentru a tine evidenta indicelui studentului afisat(utilizat strict pentru o afisare mai stilistica)
```

```
    exista_profesori boolean; -- Variabila ce primeste ca valoare rezultatul apelarii functiei afiseaza_profesori() pe un cod de student din cursorul studenti declarat in antetul pachetului
```

```
begin
```

```
    dbms_output.put_line('Studenti:');
```

```
    dbms_output.new_line;
```

```
    dbms_output.put_line('Forma afisarii: <<nume>> <<prenume>> (<<cod>>)');
```

```
    dbms_output.put_line('Profesori: <<nume>> <<prenume>> <<salariu>>');
```

```
    dbms_output.new_line;
```

```
    for i in studenti loop -- Parcurgem liniile din cursorul studenti
```

```

        dbms_output.put_line(contor || '.' || i.nume || '' || i.prenume || '(' || i.cod_student
|| ');''); -- Afisam indicele, numele, prenumele si codul studentului curent

        dbms_output.put('Profesori: '); -- Pregatim afisarea tuturor profesorilor studentului
        curent

        exista_profesori := afiseaza_profesori(i.cod_student); -- apelam functia
        afiseaza_profesori() pentru codul studentului curent

        -- Functia afiseaza_profesori() se va ocupa de afisare si va returna true daca studentul
        are profesori sau false daca nu are profesori

        if exista_profesori = false then -- Daca studentul nu are niciun profesor atunci afisam un
        mesaj sugestiv

            dbms_output.put_line('Studentul nu are niciun profesor');

            end if;

            dbms_output.new_line;
            dbms_output.new_line;

            contor := contor + 1; -- Incrementam contor-ul pentru a trece la urmatorul indice la
            afisare

            end loop;

        end afiseaza_detalii_studenti;
    
```

```

function afiseaza_profesori           -- Functie ce afiseaza toti profesorii unui student al carui
cod este dat ca parametru

    (c_student in student.cod_student%type)

    return boolean                   -- Functia returneaza true daca studentul are cel putin un
    profesor, false in caz contrar

    is

        aux rec_detalii_profesor;   -- Variabila de tip record rec_detalii_profesor ce
        memoreaza detaliile unui profesor
    
```

```

cnt  number := 1;          -- Variabila contor/count care incepe de la 1

suma  angajat.salariu%type := 0;  -- Variabila ce memoreaza suma salariilor profesorilor
studentului, suma este initializata cu 0

s_nume  student.nume%type;      -- Variabila ce memoreaza numele studentului al
caui cod este dat ca parametru

s_prenume  student.prenume%type;  -- Variabila ce memoreaza prenumele studentului
al carui cod este dat ca parametru

begin

--  imb.extend;

select nume, prenume      -- Selectam numele si prenumele studentului al carui cod este
dat ca parametru

into s_nume, s_prenume

from student

where cod_student = c_student;

-- Acest select nu va genera exceptia no_data_found,
-- intrucat aceasta va fi apelata prin intermediul procedurii afiseaza_detalii_studenti
-- procedura care facilitateaza apele numai pentru coduri de student existente in baza de
date

-- De asemenea, nu se va genera exceptia too_many_rows, intrucat cheia primara nu se
poate repeta in baza de date

open detalii_profesor(c_student);    -- Deschidem cursorul parametrizat declarat in
antetul pachetului si il apelam pentru codul dat ca parametru in functie

loop                                -- Acest cursor ne parcurge lista tuturor profesorilor studentului cu
codul dat ca parametru

```

```
    fetch detalii_profesor into aux;      -- Obtinem informatiile fiecarui profesor la care
studentul invata si le transmitem variabilei record aux
```

```
    exit when detalii_profesor%notfound;  -- Oprim bucla atunci cand am parcurs toata
lista profesorilor studentului al carui cod este dat ca parametru
```

```
    ind(cnt) := aux;      -- Tabloul indexat ind va retine pe pozitia cnt detalii profesorului
current
```

```
--      imb(c_student)(cnt) := ind(cnt);
```

```
    cnt := cnt + 1;      -- Incrementam cnt pentru urmatoarea iteratie
```

```
end loop;
```

```
close detalii_profesor;  -- Inchidem cursorul
```

```
if ind.count = 0 then      -- Daca tabloul indexat ind nu are niciun element, inseamna ca
studentul nu are niciun profesor
```

```
    ind.delete;      -- Dezalocam memoria alocata in tabloul indexat(pas inutil din moment
ce oricum ind.count = 0)
```

```
    return false;      -- Returnam false, intrucat studentul nu are niciun profesor
```

```
else      -- Altfel, inseamna ca studentul are cel putin un profesor
```

```
    for i in ind.first..ind.last loop  -- Parcurgem tabloul indexat ind pentru a obtine
informatiile pentru fiecare profesor in parte
```

```
        dbms_output.put(ind(i).nume || ' ' || ind(i).prenume || ' ' || ind(i).salariu);  --
Afisam numele, prenumele si salariul profesorului curent la care studentul invata
```

```
        suma := suma + ind(i).salariu;      -- Calculam salariul profesoral cumulativ pentru
studentul dat ca parametru
```

```
        -- Acest lucru ne va folosi pentru determinarea salariului
profesoral cumulativ maximal
```

```
        -- Salariu cumulativ = suma tuturor salariilor profesorilor din
tabloului ind
```

```
    if i = ind.last then      -- Daca am ajuns la ultimul profesor
```

```

        dbms_output.put_line('.'); -- Printam punct

    else

        dbms_output.put(', '); -- Altfel separam profesorii prin virgula

    end if; -- Observam ca nu mai este nevoie sa folosim dbms_output.new_line pentru
a se printa toate virgulele din dbms_output.put()

    end loop; -- intrucat oricum la sfarsit se va printa punctul prin dbms_output.put_line()

dbms_output.put_line('Salariu Cumulativ: ' || suma); -- Afisam salariul cumulativ
calculat insumand salariul tuturor profesorilor din ind

    if suma > maxim.salariu or -- Daca salariul cumulativ este mai mare decat salariul
cumulativ maximal

        (suma = maxim.salariu and (s_nume < maxim.nume)) or -- sau daca salariile sunt egale,
dar numele studentului dat ca parametru este mai mic alfabetic decat numele studentului cu
salariu cumulativ maximal

            (suma = maxim.salariu and s_nume = maxim.nume and s_prenume <
maxim.prenume)then -- sau daca atat salariile cumulative, cat si numele sunt egale iar
prenumele studentului cu codul dat este mai mic alfabetic decat prenumele studentului cu
salariu cumulativ maximal

                maxim.salariu := suma; -- atunci updatam salariul cumulativ maximal

                maxim.nume := s_nume; -- precum si numele

                maxim.prenume := s_prenume; -- prenumele

                maxim.cod := c_student; -- si codul studentului posesor al acestui salariu
cumulativ maximal

end if;

-- ind_salarii(c_student) := suma;

suma := 0; -- la sfarsit reinitializam suma cu 0, intrucat aceasta va fi folosita de
urmatorul student dat ca parametru

ind.delete; -- dezalocam tabloul de profesori, intrucat si acesta va fi folosit de
urmatorul student dat ca parametru

```

```
    return true; -- returnam true, intrucat am gasit cel putin un profesor pentru
studentul dat ca parametru
```

```
end if;
```

```
end afiseaza_profesori;
```

```
function salariu_cumulativ_maxim return student.cod_student%type -- Functie ce afiseaza
detaliile studentului posesor al salariului cumulativ maximal
```

```
is -- si returneaza codul acestui student
```

```
begin
```

```
    -- Afisam numele, prenumele si codul studentului posesor al salariului profesoral cumulativ
maximal
```

```
    -- Aceste detalii se afla in variabila record "maxim", declarata in antetul pachetului
```

```
    -- Iar detalii din cadrul acestei variabile au fost calculate in functia afiseaza_profesori()
prin intermediul procedurii afiseaza_detalii_studenti()
```

```
    dbms_output.put_line('Prin urmare, studentul al carui salariu cumulativ este maxim este:');
```

```
    dbms_output.put_line(maxim.nume || ' ' || maxim.prenume || '(' || maxim.cod || ')');
```

```
    dbms_output.put_line('Cu salariul cumulativ de: ' || maxim.salariu);
```

```
    dbms_output.new_line;
```

```
    dbms_output.new_line;
```

```
    return maxim.cod; -- Returnam codul studentului posesor al salariului profesoral
cumulativ maximal
```

```
end salariu_cumulativ_maxim;
```

```

function tara_facultate          -- Functie ce afiseaza tara primei facultati la care este
inscris

    (c_student in student.cod_student%type)  -- studentul cu codul dat ca parametru

    return facultate.cod_facultate%type      -- si returneaza codul acestei facultati

    is

        nume_tara  tara.nume%type;           -- Variabila in care o sa memoram numele tarii in
        care se afla facultatea studentului cu codul dat ca parametru

        s_nume    student.nume%type;         -- Numele studentului cu codul dat ca parametru

        s_prenume student.prenume%type;     -- Prenumele studentului cu codul dat ca
        parametru

        f_nume    facultate.denumire%type;   -- Numele facultatii studentului cu codul dat ca
        parametru

        f_cod     facultate.cod_facultate%type; -- Codul facultatii principale al studentului cu
        codul dat ca parametru

begin

    select t.nume, s.nume, s.prenume, f.denumire, f.cod_facultate -- Selectam numele tarii,
    numele si prenumele studentului, numele si codul facultatii

    into nume_tara, s_nume, s_prenume, f_nume, f_cod             -- In variabilele anterior
    declarate in aceasta functie

    from tara t join locatie l on (t.cod_tara = l.cod_tara)

    join facultate f on (l.cod_locatie = f.cod_locatie)

    join studiaza st on (f.cod_facultate = st.cod_facultate)

    join student s on(st.cod_student = s.cod_student)

    where s.cod_student = c_student                                -- Acolo unde codul studentului este
    egal cu cel dat ca parametru

    and rownum = 1;                                                 -- Daca exista mai multe facultati, se va selecta
    prima la care a fost inscris studentul

    -- Acest select nu va genera exceptia no_data_found, intrucat avem garantia validitatii
    codului dat ca parametru, datorita apelarii in cascada a subprogramelor din acest pachet

    -- Afisam detaliile obtinute

```

```
    dbms_output.put_line('Studentul ' || s_nume || '' || s_prenume || ' este inscris la ' ||  
    f_nume || ', care se afla in tara: ' || nume_tara);
```

```
dbms_output.new_line; -- Afisam o noua linie pentru spatiere  
return f_cod; -- Returnam codul facultatii principale  
end tara_facultate;
```

```
procedure nota_maxima(c_facultate facultate.cod_facultate%type) -- Procedura ce  
afiseaza nota maxima obtinuta in cadrul facultatii cu codul dat ca parametru
```

is

```
    f_denumire facultate.denumire%type; -- Variabila pentru numele facultatii cu codul  
dat ca parametru
```

```
    s_nume student.nume%type; -- Variabila pentru numele studentului cu nota  
maxima din cadrul facultatii cu codul dat ca parametru
```

```
    s_prenume student.prenume%type; -- Prenumele studentului cu nota maxima din  
facultate
```

```
    a_nume angajat.nume%type; -- Numele profesorului coordonator al disciplinei in  
care s-a obtinut nota maxima
```

```
    a_prenume angajat.prenume%type; -- Prenumele profesorului anterior mentionat
```

```
    c_denumire camin.denumire%type; -- Denumirea caminului in care este cazat  
studentul ce a obtinut nota maxima
```

```
    d_denumire disciplina.denumire%type; -- Denumirea disciplinei la care s-a obtinut nota  
maxima
```

```
    e_nota examinare.nota%type; -- Nota maxima obtinuta in cadrul facultatii cu  
codul dat ca parametru
```

begin

/*

Dupa ce se afiseaza tara, sa se afiseze nota maxima obtinuta in cadrul facultatii

precum si numele materiei, numele profesorului coordonator,
 numele studentului care a obtinut nota respectiva si, daca exista,
 caminul la care acesta este cazat

```

  */
-- Selectam toate detaliile cerute si le memoram in variabilele declarate in cadrul procedurii
select f.denumire, s.nume, s.prenume, a.nume, a.prenume, c.denumire, d.denumire,
e.nota
    into f_denumire, s_nume, s_prenume, a_nume, a_prenume, c_denumire, d_denumire,
e_nota

    from facultate f join studiaza st on (f.cod_facultate = st.cod_facultate)
        join student s on (st.cod_student = s.cod_student)
        join participa p on (s.cod_student = p.cod_student)
        join disciplina d on (p.cod_disciplina = d.cod_disciplina)
        join examinare e on (d.cod_disciplina = e.cod_disciplina)
        join angajat a on (a.cod_angajat = p.cod_angajat)
        join profesor prof on (a.cod_angajat = prof.cod_angajat)
        join camin c on (s.cod_camin = c.cod_camin)

    where f.cod_facultate = c_facultate and rownum = 1 -- Selectam toate detaliile pentru
facultatea cu codul dat ca parametru, daca exista mai multe materii la care s-a obtinut nota
maxima, o selectam doar pe prima

    order by nota desc; -- Ordonam descrescator dupa nota

-- Afisam detaliile cerute

    dbms_output.put_line('Pentru facultatea "' || f_denumire || "" nota maxima este ' ||
e_nota || ' si este la materia ' || d_denumire || ', ');

    dbms_output.put_line('nota fiind obtinuta de studentul ' || s_nume || ' ' || s_prenume ||
' alaturi de profesorul coordonator ');

    dbms_output.put_line(a_nume || ' ' || a_prenume || ', studentul fiind cazat la caminul ' ||
c_denumire || '.');

    dbms_output.new_line;
  
```

```

end nota_maxima;

function main return boolean      -- In functia main vom apela toate subprogramele create
in cadrul pachetului

is

    cod_salariu_maxim      student.cod_student%type;      -- Variabila in care vom memora
codul studentului al carui salariu profesoral cumulativ este maxim

    c_facultate      facultate.cod_facultate%type;

begin

    maxim.salariu := -1;      -- Initializam primul salariu maxim ca fiind -1, astfel incat orice
salariul profesoral sa fie mai mare decat -1

    afiseaza_detalii_studenti();  -- Afisam detaliiile studentilor, aceasta procedura va apela si
functia afiseaza_profesori()

    -- Dupa ce s-au apelat procedura afiseaza_detalii_student() si functia afiseaza_profesori(),
salariul cumulativ maximal a fost calculat

    cod_salariu_maxim := salariu_cumulativ_maxim();  -- Prin urmare, apelam functia
salariu_cumulativ_maxim() pentru a afisa detaliiile studentului posesor al salariului cumulativ
maximal si returnam codul acestui student, cod pe care il memoram in cod_salariu_maxim

    c_facultate := tara_facultate(cod_salariu_maxim);  -- Dupa ce am obtinut codul studentului
cu salariul cumulativ maximal, apelam functia tara_facultate() pentru a afisa tara in care se afla
prima facultate la care este inscris studentul si returnam codul acestei facultati

    nota_maxima(c_facultate);  -- Cu codul facultatii obtinut, apelam procedura
nota_maxima() pentru a afisa nota maxima obtinuta in cadrul acelei facultati, precum si toate
detaliile legate de studentul si profesorul coordonator

    return true;      -- Functia main returneaza true atunci cand toate procedurile si functiile
s-au executat cu succes

end main;

end exercitiul_14;
/

```

-- Rezultate/Explicatii:

-- Bloc anonim in care testam pachetul definit

declare

 rezultat boolean;

begin

 rezultat := exercitiul_14.main();

end;

/

The screenshot shows the Oracle SQL Developer interface. On the left, the 'Worksheet' tab displays the PL/SQL code. Lines 263 to 269 are highlighted in blue, representing the anonymous block being executed. The code defines a variable 'rezultat' and sets it to the result of calling 'exercitiul_14.main()'. The rest of the code handles the output of student information based on their participation in various activities.

The 'Doms Output' tab on the right shows the results of the execution. It lists five students with their names, ages, professors, and cumulative salaries. The output is as follows:

1. Ionescu Stefan (11)
Profesor: Popovici Alexandra 1800, Stefanescu Teodora 2500.
Salariu Cumulativ: 4300
2. Iliescu Mihail (22)
Profesor: Stefanescu Teodora 2500.
Salariu Cumulativ: 2500
3. Georgescu Nicolae (33)
Profesor: Stanescu Constantin 3700, Stefanescu Teodora 2500.
Salariu Cumulativ: 6200
4. Emil Luca (44)
Profesor: Bogdan Octavian 5000, Mihai George 1900, Popovici Alexandra 1800.
Salariu Cumulativ: 8700
5. Florentin Daniel (55)
Profesor: Bogdan Octavian 5000, Stanescu Constantin 3700.
Salariu Cumulativ: 8700

At the bottom, the status bar indicates the line number (Line 268), column (Column 2), and date (27-Dec-21).

The screenshot shows the Oracle SQL Developer interface. In the top-left pane, there is a code editor window titled 'ex14.3.sql' containing PL/SQL code. The code includes a package body named 'EXERCITIUL_14' with a main procedure that calls a function 'nota_maxima'. The function returns true if the student's cumulative salary is the maximum. The main procedure then prints the details of the student with the highest cumulative salary. The output window on the right, titled 'Dbms Output', displays the results of the execution. It lists five students with their names, ages, professors, and cumulative salaries. It also prints a message indicating that student Emil Luca has the highest cumulative salary and is studying Medicine in Romania. Finally, it prints the total cumulative salary of 8700.

```
257 nota_maxima(c_facultate); -- Cu codul facultatii
258     return true;           -- Functia main returneaza true
259 end main;
260 end exercitiul_14;
261 /
262 -- Bloc anonim in care testam pachetul definit
263 declare
264     resultat boolean;
265 begin
266     resultat := exercitiul_14.main();
267 end;
268 /
269 /*
270 select * from participa;
271
272 select s.nume, t.nume
273 from tara t join locatie l on (t.cod_tara = l.cod_tara)
274      join student s on (l.cod_locatie = s.cod_locatie)
275      join profesor p on (s.cod_profesor = p.cod_profesor)
276      join matematica m on (p.cod_materie = m.cod_materie)
277      join facultate f on (m.cod_facultate = f.cod_facultate)
278      join facultate fm on (f.cod_facultate = fm.cod_facultate)
279      join facultate fm on (fm.cod_facultate = fm.cod_facultate)
280      join facultate fm on (fm.cod_facultate = fm.cod_facultate)
281      join facultate fm on (fm.cod_facultate = fm.cod_facultate)
282      join facultate fm on (fm.cod_facultate = fm.cod_facultate)
283      join facultate fm on (fm.cod_facultate = fm.cod_facultate)
284      join facultate fm on (fm.cod_facultate = fm.cod_facultate)
285      join facultate fm on (fm.cod_facultate = fm.cod_facultate)
286      join facultate fm on (fm.cod_facultate = fm.cod_facultate)
287      join facultate fm on (fm.cod_facultate = fm.cod_facultate)
288      join facultate fm on (fm.cod_facultate = fm.cod_facultate)
289      join facultate fm on (fm.cod_facultate = fm.cod_facultate)
290      join facultate fm on (fm.cod_facultate = fm.cod_facultate)
291      join facultate fm on (fm.cod_facultate = fm.cod_facultate)
292      join facultate fm on (fm.cod_facultate = fm.cod_facultate)
293      join facultate fm on (fm.cod_facultate = fm.cod_facultate)
294      join facultate fm on (fm.cod_facultate = fm.cod_facultate)
295      join facultate fm on (fm.cod_facultate = fm.cod_facultate)
296      join facultate fm on (fm.cod_facultate = fm.cod_facultate)
297      join facultate fm on (fm.cod_facultate = fm.cod_facultate)
298      join facultate fm on (fm.cod_facultate = fm.cod_facultate)
299      join facultate fm on (fm.cod_facultate = fm.cod_facultate)
299
3. Georgescu Nicolae (33)
Profesori: Stanescu Constantin 3700, Stefanescu Teodora 2500.
Salariu Cumulativ: 6200

4. Emil Luca (44)
Profesori: Bogdan Octavian 5000, Mihai George 1900, Popovici Alexandra 1800.
Salariu Cumulativ: 8700

5. Florentin Daniel (55)
Profesori: Bogdan Octavian 5000, Stanescu Constantin 3700.
Salariu Cumulativ: 8700

Prin urmare, studentul al carui salariu cumulativ este maxim este:
Emil Luca (44)
Cu salariu cumulativ de: 8700

Studentul Emil Luca este inscris la Facultatea de Medicina, care se afla in tara: SUA

Pentru facultatea "Facultatea de Medicina" nota maxima este 10 si este la materia Matematica,
nota fiind obtinuta de studentul Georgescu Nicolae alaturi de profesorul coordonator
Stefanescu Teodora, studentul fiind cazat la caminul Magic Dorm.
```

-- Toate explicatiile necesare se regasesc in cadrul dbms_output din cele doua poze

