**­­­­PROIECT SGBD**

**GESTIONAREA UNEI FACULTĂȚI**

**-Chițu Tudor 231 –**

Contents

[**1.** **Prezentați pe scurt baza de date** 3](#_Toc155980073)

[Constrângerile asupra modelului sunt: 3](#_Toc155980074)

[**2.** **Realizați diagrama entitate-relație.** 3](#_Toc155980075)

[**3.** **Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare.** 4](#_Toc155980076)

[Atributele entităților prin intermediul schemelor relaționale: 5](#_Toc155980077)

[**4.** **Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, implementând toate constrângerile de integritate necesare (chei primare, cheile externe etc).** 6](#_Toc155980078)

[Table STUDENT 6](#_Toc155980079)

[Table TAXA 6](#_Toc155980080)

[Table BUGET 6](#_Toc155980081)

[Table ERASMUS 7](#_Toc155980082)

[Table TARI 7](#_Toc155980083)

[Table STUDII\_STRAINATATE 7](#_Toc155980084)

[Table CAZ\_SOCIAL 8](#_Toc155980085)

[Table BURSA 8](#_Toc155980086)

[Table LICENTA 8](#_Toc155980087)

[Table SPECIALIZARE 9](#_Toc155980088)

[Table PROFESOR 9](#_Toc155980089)

[Table CURS 10](#_Toc155980090)

[Table DECANAT 10](#_Toc155980091)

[Table RELATIE\_SPECIALIZARE\_CURS 10](#_Toc155980092)

[Table RELATIE\_STUDENT\_CURS 10](#_Toc155980093)

[**5.** **Adăugați informații coerente în tabelele create.** 11](#_Toc155980094)

[**6.** **Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze toate cele 3 tipuri de colecții studiate. Apelați subprogramul.** 15](#_Toc155980095)

[Cerința 15](#_Toc155980096)

[Rezolvarea 15](#_Toc155980097)

[**7.** **Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze 2 tipuri diferite de cursoare studiate, unul dintre acestea fiind cursor parametrizat, dependent de celălalt cursor. Apelați subprogamul.** 20](#_Toc155980098)

[Cerința 20](#_Toc155980099)

[Rezolvarea 20](#_Toc155980100)

[**8.** **Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Definiți minim 2 excepții proprii. Apelați subprogramul astfel încât să evidențiați toate cazurile definite și tratate.** 23](#_Toc155980101)

[Cerința 23](#_Toc155980102)

[Rezolvarea 23](#_Toc155980103)

[**9.** **Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să utilizeze într-o singură comandă SQL 5 dintre tabelele definite. Tratați toate excepțiile care pot apărea, incluzând excepțiie NO\_DATA\_FOUND și TOO\_MANY\_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.** 27](#_Toc155980104)

[Cerința 27](#_Toc155980105)

[Rezolvarea 27](#_Toc155980106)

[**10.** **11. Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul. Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.** 31](#_Toc155980107)

[Cerința 31](#_Toc155980108)

[Rezolvarea 31](#_Toc155980109)

[**11.** **Rezolvat împreună cu 10.** 33](#_Toc155980110)

[**12.** **Definiți un trigger de tip LDD. Declanșați trigger-ul.** 33](#_Toc155980111)

[Cerința 33](#_Toc155980112)

[Rezolvarea 33](#_Toc155980113)

[**13.** **Definiți un pachet care să conțină toate obiectele definite în cadrul proiectului.** 35](#_Toc155980114)

[Rezolvarea 35](#_Toc155980115)

[**14.** **Definiți un pachet care să includă tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date definite (minim 2 tipuri de date, minim 2 funcții, minim 2 proceduri).** 51](#_Toc155980116)

[Cerința 51](#_Toc155980117)

[Rezolvarea 51](#_Toc155980118)

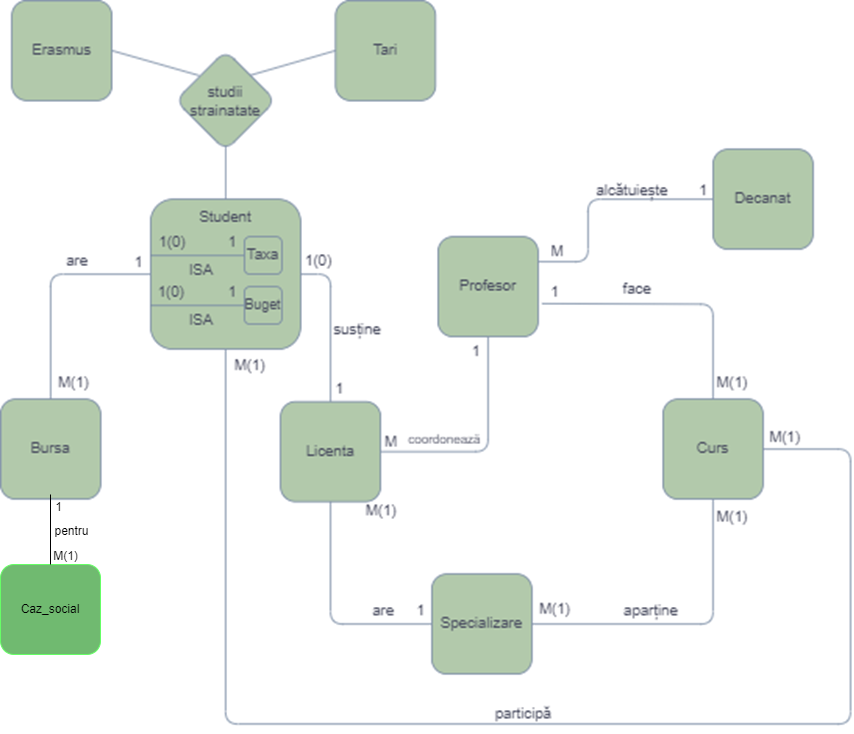
# **Prezentați pe scurt baza de date**

Gestionarea facultății urmărește situația studenților cu privire la grupa din care fac parte, anul de studiu, cnp-ul, numele și prenumele. De asemenea urmărește repartizarea acestora la taxa sau buget, dar și cei care sunt bursieri. Unii dintre ei lucrează și la o lucrare de licență, având un profesor îndrumător. Cursurile la facultate sunt diverse, specifice specializărilor și ținute de un anumit profesor. Facultatea are Decanat, format din profesori. Facultatea are parteneriate Erasmus în diferite țări.

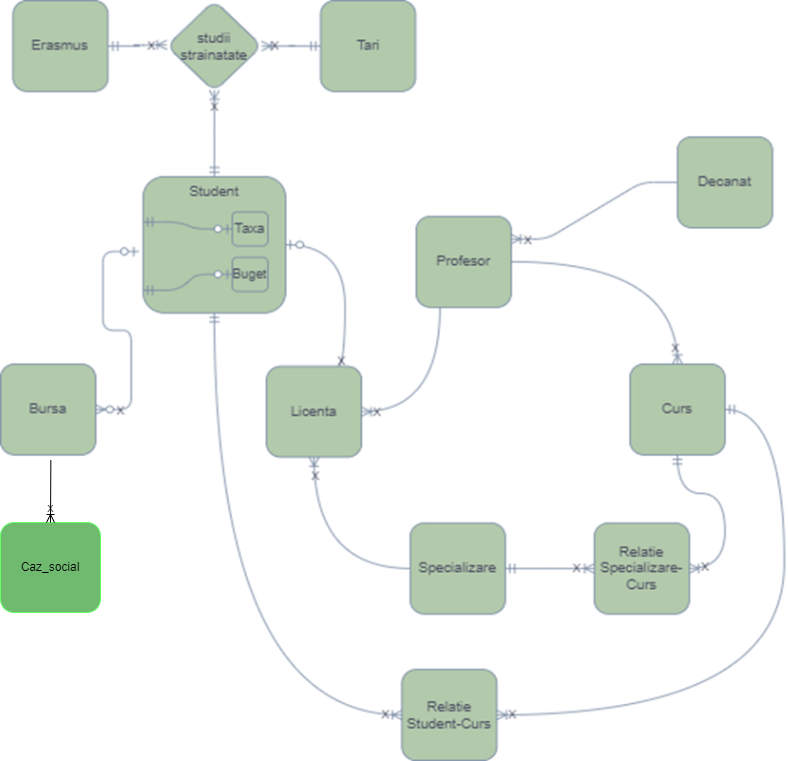
## Constrângerile asupra modelului sunt:

* Un student poate fi ori la buget, ori la taxă
* Grupa în care poate fi un student este un număr între 100 și 500
* Specializările sunt:Informatică, Matematică, Matematică-Informatică, Matematică-Aplicată, CTI
* Licența trebuie să fie susținută de un fix un student, cu un singur profesor îndrumător și o singură arie de studiu. După absolvirea studentului, lucrarea sa de licență păstrează doar numele și id-ul acesteia
* Decanatul actual are minim un membru, iar baza de date reține și cronologia decanatului
* Fiecare curs are un doar profesor
* Numele lucrării de licență este unic

# **Realizați diagrama entitate-relație.**



# **Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare.**



## Atributele entităților prin intermediul schemelor relaționale:

Bursă(id\_bursa (PK), nume, suma, id\_student (FK))

Student(id\_ student(PK), nume, prenume, cnp, an\_studiu, grupa, tip\_student)

Taxă(id\_student (PK,FK), valoare\_taxa)

Buget(id\_student (PK,FK), medie\_admitere)

Licență(id\_licenta (PK), nume, id\_student (FK), id\_profesor (FK), arie\_subiect (FK))

Relație\_Student\_Curs(id\_relatie\_sc (PK), id\_student (FK), id\_curs (FK))

Curs(id\_curs (PK), nume, id\_specializare (FK), id\_profesor (FK))

Profesor(id\_profesor (PK), nume, prenume, cnp, id\_specializare (FK), vechime, id\_decanat (FK))

Decanat(id\_decanat (PK), data, durata)

Specializare(id\_specializare (PK), nume, nume\_diploma)

Relație\_Specializare\_Curs(id\_relatie\_spc (PK), id\_specializare (FK), id\_curs (FK))

Tari(id\_tara (PK), nume, moneda, limba)

Erasmus(id\_erasmus (PK), nume, durata, data\_inceput)

Studii\_strainatate(id\_relatie\_ss (PK), id\_tara (FK), id\_erasmus (FK), id\_student (FK))

Caz\_social (id\_caz\_social (PK), nr\_adeverinte, venit\_lunar, id\_bursa (FK))

# **Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, implementând toate constrângerile de integritate necesare (chei primare, cheile externe etc).**

## Table STUDENT

CREATE TABLE "TEDI"."STUDENT"

( "ID\_STUDENT" NUMBER(5,0),

"NUME" VARCHAR2(30 BYTE),

"PRENUME" VARCHAR2(30 BYTE),

"CNP" VARCHAR2(16 BYTE),

"AN\_STUDIU" NUMBER(1,0),

"GRUPA" NUMBER(3,0),

"TIP\_STUDENT" VARCHAR2(20 BYTE)

)

ALTER TABLE "TEDI"."STUDENT" MODIFY ("NUME" NOT NULL ENABLE);

ALTER TABLE "TEDI"."STUDENT" MODIFY ("PRENUME" NOT NULL ENABLE);

ALTER TABLE "TEDI"."STUDENT" MODIFY ("CNP" NOT NULL ENABLE);

ALTER TABLE "TEDI"."STUDENT" MODIFY ("AN\_STUDIU" NOT NULL ENABLE);

ALTER TABLE "TEDI"."STUDENT" MODIFY ("GRUPA" NOT NULL ENABLE);

ALTER TABLE "TEDI"."STUDENT" MODIFY ("TIP\_STUDENT" NOT NULL ENABLE);

ALTER TABLE "TEDI"."STUDENT" ADD PRIMARY KEY ("ID\_STUDENT")

ALTER TABLE "TEDI"."STUDENT" ADD CONSTRAINT "GRUPA\_3" CHECK (grupa between 100 and 500) ENABLE;

## Table TAXA

CREATE TABLE "TEDI"."TAXA"

( "ID\_STUDENT" NUMBER(5,0),

"VALOARE\_TAXA" NUMBER(4,0)

)

ALTER TABLE "TEDI"."TAXA" MODIFY ("VALOARE\_TAXA" NOT NULL ENABLE);

ALTER TABLE "TEDI"."TAXA" ADD PRIMARY KEY ("ID\_STUDENT")

ALTER TABLE "TEDI"."TAXA" ADD FOREIGN KEY ("ID\_STUDENT")

REFERENCES "TEDI"."STUDENT" ("ID\_STUDENT") ENABLE;

## Table BUGET

CREATE TABLE "TEDI"."BUGET"

( "ID\_STUDENT" NUMBER(5,0),

"MEDIE\_ADMITERE" NUMBER(4,2)

)

ALTER TABLE "TEDI"."BUGET" MODIFY ("MEDIE\_ADMITERE" NOT NULL ENABLE);

ALTER TABLE "TEDI"."BUGET" ADD PRIMARY KEY ("ID\_STUDENT")

ALTER TABLE "TEDI"."BUGET" ADD FOREIGN KEY ("ID\_STUDENT")

REFERENCES "TEDI"."STUDENT" ("ID\_STUDENT") ENABLE;

## Table ERASMUS

CREATE TABLE "TEDI"."ERASMUS"

( "ID\_ERASMUS" NUMBER(5,0),

"NUME" VARCHAR2(30 BYTE),

"DURATA" NUMBER(5,0),

"DATA\_INCEPUT" DATE

)

ALTER TABLE "TEDI"."ERASMUS" MODIFY ("NUME" NOT NULL ENABLE);

ALTER TABLE "TEDI"."ERASMUS" MODIFY ("DURATA" NOT NULL ENABLE);

ALTER TABLE "TEDI"."ERASMUS" MODIFY ("DATA\_INCEPUT" NOT NULL ENABLE);

ALTER TABLE "TEDI"."ERASMUS" ADD PRIMARY KEY ("ID\_ERASMUS")

## Table TARI

CREATE TABLE "TEDI"."TARI"

( "ID\_TARA" NUMBER(5,0),

"NUME" VARCHAR2(30 BYTE),

"MONEDA" VARCHAR2(3 BYTE),

"LIMBA" VARCHAR2(30 BYTE)

)

ALTER TABLE "TEDI"."TARI" MODIFY ("NUME" NOT NULL ENABLE);

ALTER TABLE "TEDI"."TARI" MODIFY ("MONEDA" NOT NULL ENABLE);

ALTER TABLE "TEDI"."TARI" MODIFY ("LIMBA" NOT NULL ENABLE);

ALTER TABLE "TEDI"."TARI" ADD PRIMARY KEY ("ID\_TARA")

## Table STUDII\_STRAINATATE

CREATE TABLE "TEDI"."STUDII\_STRAINATATE"

( "ID\_RELATIE\_SS" NUMBER(5,0),

"ID\_STUDENT" NUMBER(5,0),

"ID\_TARA" NUMBER(5,0),

"ID\_ERASMUS" NUMBER(5,0)

)

ALTER TABLE "TEDI"."STUDII\_STRAINATATE" MODIFY ("ID\_STUDENT" NOT NULL ENABLE);

ALTER TABLE "TEDI"."STUDII\_STRAINATATE" MODIFY ("ID\_TARA" NOT NULL ENABLE);

ALTER TABLE "TEDI"."STUDII\_STRAINATATE" MODIFY ("ID\_ERASMUS" NOT NULL ENABLE);

ALTER TABLE "TEDI"."STUDII\_STRAINATATE" ADD PRIMARY KEY ("ID\_RELATIE\_SS")

ALTER TABLE "TEDI"."STUDII\_STRAINATATE" ADD FOREIGN KEY ("ID\_STUDENT")

REFERENCES "TEDI"."STUDENT" ("ID\_STUDENT") ENABLE;

ALTER TABLE "TEDI"."STUDII\_STRAINATATE" ADD FOREIGN KEY ("ID\_ERASMUS")

REFERENCES "TEDI"."ERASMUS" ("ID\_ERASMUS") ENABLE;

ALTER TABLE "TEDI"."STUDII\_STRAINATATE" ADD FOREIGN KEY ("ID\_TARA")

REFERENCES "TEDI"."TARI" ("ID\_TARA") ENABLE;

## Table CAZ\_SOCIAL

CREATE TABLE "TEDI"."CAZ\_SOCIAL"

( "ID\_CAZ\_SOCIAL" NUMBER(5,0),

"NR\_ADEVERINTE" NUMBER(5,0),

"VENIT\_LUNAR" NUMBER(5,0),

"ID\_BURSA" NUMBER(5,0)

)

ALTER TABLE "TEDI"."CAZ\_SOCIAL" ADD PRIMARY KEY ("ID\_CAZ\_SOCIAL")

ALTER TABLE "TEDI"."CAZ\_SOCIAL" ADD FOREIGN KEY ("ID\_BURSA")

REFERENCES "TEDI"."BURSA" ("ID\_BURSA") ENABLE;

## Table BURSA

CREATE TABLE "TEDI"."BURSA"

( "ID\_BURSA" NUMBER(5,0),

"NUME" VARCHAR2(30 BYTE),

"SUMA" NUMBER(4,0),

"ID\_STUDENT" NUMBER(5,0)

)

ALTER TABLE "TEDI"."BURSA" ADD CONSTRAINT "SUMA\_MINIMA" CHECK (suma >= 500) ENABLE;

ALTER TABLE "TEDI"."BURSA" ADD CONSTRAINT "SUMA\_MAXIMA" CHECK (suma <=1500) ENABLE;

ALTER TABLE "TEDI"."BURSA" MODIFY ("NUME" NOT NULL ENABLE);

ALTER TABLE "TEDI"."BURSA" MODIFY ("SUMA" NOT NULL ENABLE);

ALTER TABLE "TEDI"."BURSA" MODIFY ("ID\_STUDENT" NOT NULL ENABLE);

ALTER TABLE "TEDI"."BURSA" ADD PRIMARY KEY ("ID\_BURSA")

ALTER TABLE "TEDI"."BURSA" ADD FOREIGN KEY ("ID\_STUDENT")

REFERENCES "TEDI"."STUDENT" ("ID\_STUDENT") ENABLE;

## Table LICENTA

CREATE TABLE "TEDI"."LICENTA"

( "ID\_LICENTA" NUMBER(5,0),

"NUME" VARCHAR2(30 BYTE),

"ID\_STUDENT" NUMBER(5,0),

"ID\_SPECIALIZARE" NUMBER(5,0),

"ID\_PROFESOR" NUMBER(5,0)

)

ALTER TABLE "TEDI"."LICENTA" MODIFY ("NUME" NOT NULL ENABLE);

ALTER TABLE "TEDI"."LICENTA" ADD PRIMARY KEY ("ID\_LICENTA")

ALTER TABLE "TEDI"."LICENTA" ADD UNIQUE ("NUME")

ALTER TABLE "TEDI"."LICENTA" ADD FOREIGN KEY ("ID\_STUDENT")

REFERENCES "TEDI"."STUDENT" ("ID\_STUDENT") ENABLE;

ALTER TABLE "TEDI"."LICENTA" ADD FOREIGN KEY ("ID\_PROFESOR")

REFERENCES "TEDI"."PROFESOR" ("ID\_PROFESOR") ENABLE;

ALTER TABLE "TEDI"."LICENTA" ADD FOREIGN KEY ("ID\_SPECIALIZARE")

REFERENCES "TEDI"."SPECIALIZARE" ("ID\_SPECIALIZARE") ENABLE;

## Table SPECIALIZARE

CREATE TABLE "TEDI"."SPECIALIZARE"

( "ID\_SPECIALIZARE" NUMBER(5,0),

"NUME" VARCHAR2(30 BYTE),

"NUME\_DIPLOMA" VARCHAR2(60 BYTE)

)

ALTER TABLE "TEDI"."SPECIALIZARE" MODIFY ("NUME" NOT NULL ENABLE);

ALTER TABLE "TEDI"."SPECIALIZARE" MODIFY ("NUME\_DIPLOMA" NOT NULL ENABLE);

ALTER TABLE "TEDI"."SPECIALIZARE" ADD PRIMARY KEY ("ID\_SPECIALIZARE")

ALTER TABLE "TEDI"."SPECIALIZARE" ADD CONSTRAINT "NUME\_SPECIALIZARE" CHECK (nume = 'Matematica Pura' or nume = 'CTI' or nume = 'Informatica' or

nume = 'Matematica Informatica' or nume = 'Matematica Aplicata') ENABLE;

## Table PROFESOR

CREATE TABLE "TEDI"."PROFESOR"

( "ID\_PROFESOR" NUMBER,

"NUME" VARCHAR2(30 BYTE),

"PRENUME" VARCHAR2(30 BYTE),

"CNP" VARCHAR2(16 BYTE),

"VECHIME" NUMBER(2,0),

"ID\_DECANAT" NUMBER(5,0)

)

ALTER TABLE "TEDI"."PROFESOR" MODIFY ("NUME" NOT NULL ENABLE);

ALTER TABLE "TEDI"."PROFESOR" MODIFY ("PRENUME" NOT NULL ENABLE);

ALTER TABLE "TEDI"."PROFESOR" ADD PRIMARY KEY ("ID\_PROFESOR")

ALTER TABLE "TEDI"."PROFESOR" ADD FOREIGN KEY ("ID\_DECANAT")

REFERENCES "TEDI"."DECANAT" ("ID\_DECANAT") ENABLE;

## Table CURS

CREATE TABLE "TEDI"."CURS"

( "ID\_CURS" NUMBER(5,0),

"NUME" VARCHAR2(30 BYTE),

"ID\_PROFESOR" NUMBER(5,0)

)

ALTER TABLE "TEDI"."CURS" MODIFY ("NUME" NOT NULL ENABLE);

ALTER TABLE "TEDI"."CURS" MODIFY ("ID\_PROFESOR" NOT NULL ENABLE);

ALTER TABLE "TEDI"."CURS" ADD PRIMARY KEY ("ID\_CURS")

ALTER TABLE "TEDI"."CURS" ADD FOREIGN KEY ("ID\_PROFESOR")

REFERENCES "TEDI"."PROFESOR" ("ID\_PROFESOR") ENABLE;

## Table DECANAT

CREATE TABLE "TEDI"."DECANAT"

( "ID\_DECANAT" NUMBER,

"DATA" DATE,

"DURATA" NUMBER

)

ALTER TABLE "TEDI"."DECANAT" ADD PRIMARY KEY ("ID\_DECANAT")

## Table RELATIE\_SPECIALIZARE\_CURS

CREATE TABLE "TEDI"."RELATIE\_SPECIALIZARE\_CURS"

( "ID\_RELATIE\_SPC" NUMBER(5,0),

"ID\_SPECIALIZARE" NUMBER(5,0),

"ID\_CURS" NUMBER(5,0)

)

ALTER TABLE "TEDI"."RELATIE\_SPECIALIZARE\_CURS" ADD PRIMARY KEY ("ID\_RELATIE\_SPC")

ALTER TABLE "TEDI"."RELATIE\_SPECIALIZARE\_CURS" ADD FOREIGN KEY ("ID\_CURS")

REFERENCES "TEDI"."CURS" ("ID\_CURS") ENABLE;

ALTER TABLE "TEDI"."RELATIE\_SPECIALIZARE\_CURS" ADD FOREIGN KEY ("ID\_SPECIALIZARE")

REFERENCES "TEDI"."SPECIALIZARE" ("ID\_SPECIALIZARE") ENABLE;

## Table RELATIE\_STUDENT\_CURS

CREATE TABLE "TEDI"."RELATIE\_STUDENT\_CURS"

( "ID\_RELATIE\_SC" NUMBER(5,0),

"ID\_STUDENT" NUMBER(5,0),

"ID\_CURS" NUMBER(5,0)

)

ALTER TABLE "TEDI"."RELATIE\_STUDENT\_CURS" ADD PRIMARY KEY ("ID\_RELATIE\_SC")

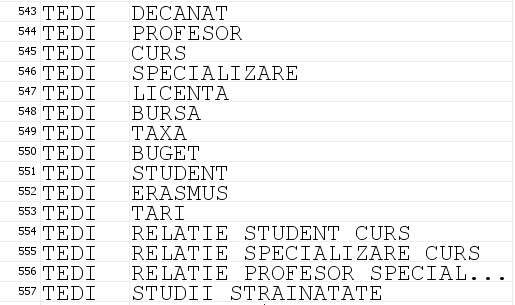
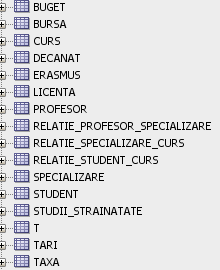
ALTER TABLE "TEDI"."RELATIE\_STUDENT\_CURS" ADD FOREIGN KEY ("ID\_STUDENT")

REFERENCES "TEDI"."STUDENT" ("ID\_STUDENT") ENABLE;

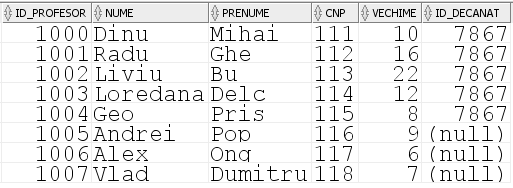
ALTER TABLE "TEDI"."RELATIE\_STUDENT\_CURS" ADD FOREIGN KEY ("ID\_CURS")

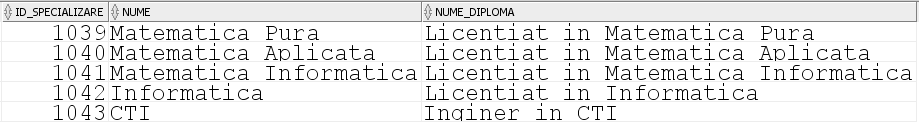
REFERENCES "TEDI"."CURS" ("ID\_CURS") ENABLE;

# **Adăugați informații coerente în tabelele create.**

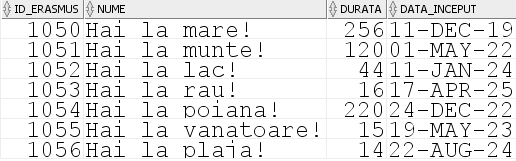


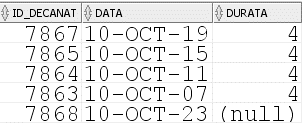
Inserarea în tabele:

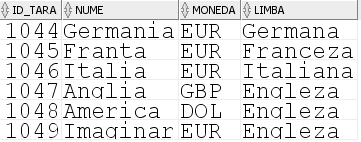


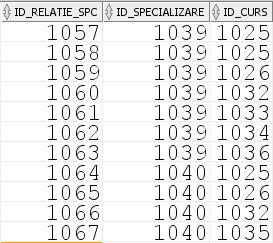
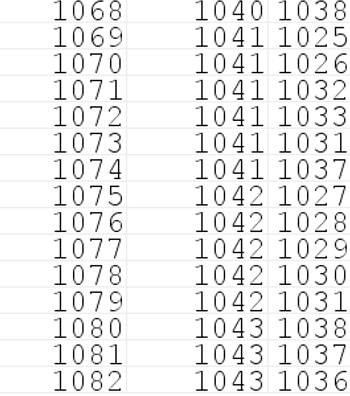


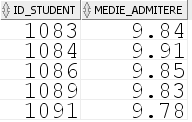
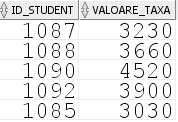


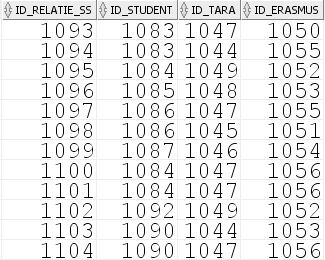
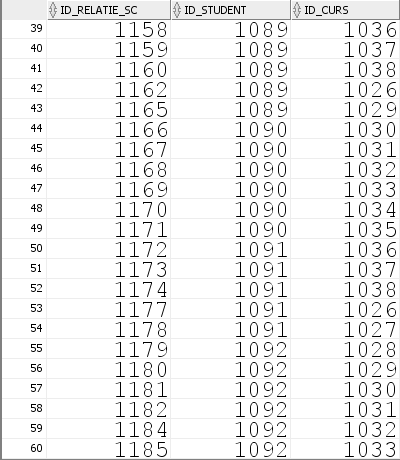


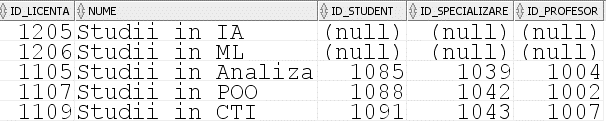


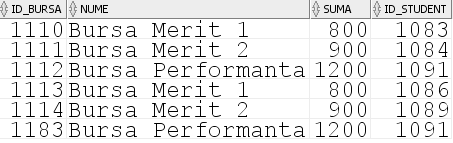


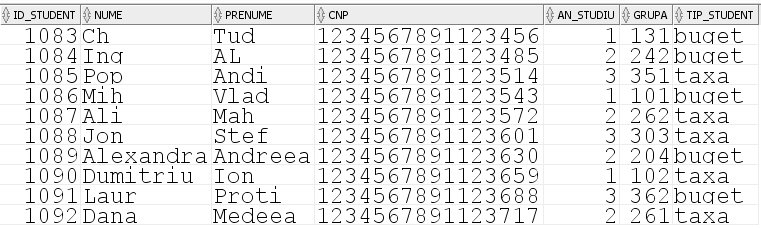












# **Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze toate cele 3 tipuri de colecții studiate. Apelați subprogramul.**

## Cerința

Un student se numește nActiv dacă are cel mai mic numar de burse sociale care au mai mult de n adeverinte. Daca sunt mai multe astfel de burse, le ordonăm după valoare. Dorim să aflăm numele și id-ul studenților nActivi.

## Rezolvarea

CREATE OR REPLACE PROCEDURE findMostNActiveStudent

(v\_nrcomp NUMBER DEFAULT 1)

IS

/\*Record\*/

type student\_info is record (

id\_student STUDENT.id\_student%TYPE,

nume\_student STUDENT.nume%TYPE,

id\_bursa BURSA.id\_bursa%TYPE

);

/\*Tablou imbricat\*/

type tablou\_imbricat is table of student\_info;

v\_list\_students tablou\_imbricat := tablou\_imbricat();

v\_old\_student STUDENT.id\_student%TYPE;

v\_nume STUDENT.nume%TYPE;

/\*Tablou indexat\*/

type tablou\_indexat is table of bursa.id\_bursa%TYPE index by pls\_integer;

v\_list\_burse tablou\_indexat;

v\_allburse number;

/\*Vector\*/

type vector is varray(150) of STUDENT.id\_student%TYPE;

v\_list\_studentids vector := vector();

i number;

j number;

minim number;

contor number;

too\_many\_burse exception;

too\_many\_burse\_cautate exception;

BEGIN

select count(\*)

into v\_allburse

from bursa;

if(v\_nrcomp > v\_allburse) then

raise too\_many\_burse;

end if;

select BAUX.ID\_BURSA

bulk collect into v\_list\_burse

from BURSA BAUX

join (select B.id\_bursa, count(\*) as nr\_burse\_sociale

from BURSA B

join caz\_social C2 on C2.id\_bursa = B.id\_bursa

where lower(C2.nr\_adeverinte) > 2

GROUP BY B.id\_bursa) D on D.id\_bursa = BAUX.id\_bursa

order by D.nr\_burse\_sociale desc, BAUX.suma;

if(v\_nrcomp > v\_list\_burse.COUNT) then

raise too\_many\_burse\_cautate;

end if;

select s2.id\_student, s2.nume, b.id\_bursa

bulk collect into v\_list\_students

from bursa b

join student s2 on s2.id\_student = b.id\_student

order by s2.id\_student;

v\_old\_student := v\_list\_students(1).id\_student;

contor := 0;

minim := v\_list\_students.count;

for i in v\_list\_students.first..v\_list\_students.last loop

if(v\_old\_student <> v\_list\_students(i).id\_student) then

if(minim = contor and contor <> 0) then

v\_list\_studentids.extend;

v\_list\_studentids(v\_list\_studentids.last) := v\_old\_student;

elsif(minim > contor and contor <> 0) then

v\_list\_studentids.delete;

v\_list\_studentids.extend;

v\_list\_studentids(v\_list\_studentids.last) := v\_old\_student;

minim := contor;

end if;

v\_old\_student := v\_list\_students(i).id\_student;

contor:= 0;

end if;

for j in 1..v\_nrcomp loop

if(v\_list\_students(i).id\_bursa = v\_list\_burse(j)) then

contor := contor + 1;

end if;

end loop;

end loop;

if(minim = contor and contor <> 0) then

v\_list\_studentids.extend;

v\_list\_studentids(v\_list\_studentids.last) := v\_old\_student;

elsif(minim > contor and contor <> 0) then

v\_list\_studentids.delete;

v\_list\_studentids.extend;

v\_list\_studentids(v\_list\_studentids.last) := v\_old\_student;

minim := contor;

end if;

for i in v\_list\_studentids.first..v\_list\_studentids.last loop

select nume

into v\_nume

from student

where id\_student = v\_list\_studentids(i);

if(v\_nume is not null) then

DBMS\_OUTPUT.PUT\_LINE('Student ' || v\_nrcomp || '-activ: id = ' || v\_list\_studentids(i) || ' nume = '

|| v\_nume);

else

DBMS\_OUTPUT.PUT\_LINE('Student ' || v\_nrcomp || '-activ: id = ' || v\_list\_studentids(i) || ' fara

nume');

end if;

end loop;

EXCEPTION

when too\_many\_burse then

RAISE\_APPLICATION\_ERROR(-20001,'Prea putine burse cerute');

when too\_many\_burse\_cautate then

RAISE\_APPLICATION\_ERROR(-20002,'Prea multe acte cerute');

when TOO\_MANY\_ROWS then

RAISE\_APPLICATION\_ERROR(-20003,'Prea multe burse gasite cu acelasi id');

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20004,'Nu s-au gasit datele');

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20000,'Alta eroare!');

END findMostNActiveStudent;

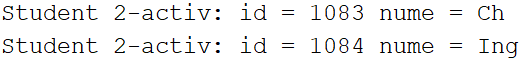
/

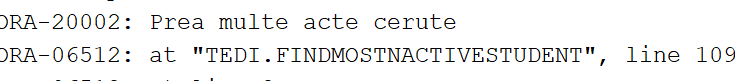
begin

findMostNActiveStudent(2);

end;

/







# **Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze 2 tipuri diferite de cursoare studiate, unul dintre acestea fiind cursor parametrizat, dependent de celălalt cursor. Apelați subprogramul.**

## Cerința

Din cauza scumpirii prețurilor de călătorie, Universitatea dorește să majoreze bursele unor studenți. Fondul alocat nu este suficient de mare pentru toți, așa că Universitatea a decis să mărească bursele studenților care sunt plecați pe o durata de mai mult de 50 de zile și au cel mult N burse. Majorarea acestora va avea loc doar pentru bursele de merit, cu un procent de 20%.

## Rezolvarea

CREATE OR REPLACE PROCEDURE proc1

(v\_in NUMBER DEFAULT 0)

IS

contor number;

flag number := 0;

cursor c is

select s.id\_student, count(\*) nr\_burse

from STUDENT s

join bursa B2 on s.id\_student = B2.id\_student

group by s.id\_student

having count(\*) <= v\_in;

cursor u(v\_idstudent STUDENT.id\_student%type) is

select \*

from bursa

where id\_student = v\_idstudent and id\_student in (

select id\_student

from studii\_strainatate

join ERASMUS E3 on E3.id\_erasmus = studii\_strainatate.id\_erasmus

where E3.durata >= 100

)

for update of suma nowait ;

no\_students exception;

BEGIN

for student in c loop

flag := 1;

contor := 0;

for j in u(student.id\_student) loop

if(lower(j.nume) like 'bursa merit%') then

contor := contor + 1;

update BURSA

set suma = suma\*1.2

where current of u;

end if;

end loop;

DBMS\_OUTPUT.PUT('Studentul '|| student.id\_student);

if(contor = 0) then

DBMS\_OUTPUT.PUT\_LINE(' nu si-a marit burse de merit');

else

DBMS\_OUTPUT.PUT\_LINE(' si-a marit '|| contor || ' burse');

end if;

rollback; -------------------------------

end loop;

if(flag = 0) then

raise no\_students;

end if;

EXCEPTION

when INVALID\_CURSOR then

RAISE\_APPLICATION\_ERROR(-20001, 'Cursorul este inchis');

when CURSOR\_ALREADY\_OPEN then

RAISE\_APPLICATION\_ERROR(-20002, 'Cursor deja deschis');

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20003,'Nu s-au gasit date');

WHEN no\_students THEN

RAISE\_APPLICATION\_ERROR(-20004,'Nu sunt studenti cu acest nr minim de burse');

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20000,'Alta eroare!');

END proc1;

/

DECLARE

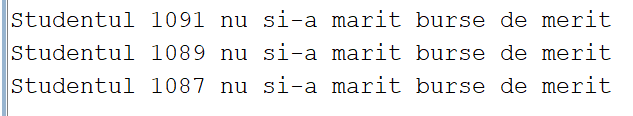
v\_x number := &p;

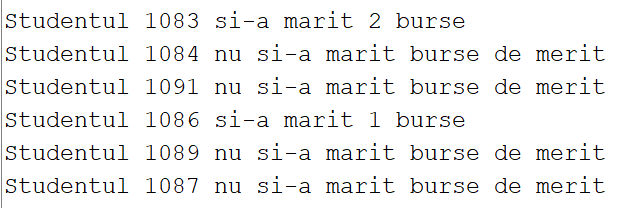
BEGIN

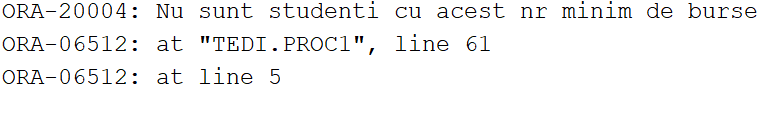
proc1(v\_x);

END;

/







# **Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Definiți minim 2 excepții proprii. Apelați subprogramul astfel încât să evidențiați toate cazurile definite și tratate.**

## Cerința

În urma promovării unor examene de limbă străină, unii studenți au reușit să meargă într-un proiect Erasmus. Universitatea dorește să acrediteze aceste deplasări, fiind nevoie de o actualizare a bazei de date a studenților, pentru aceștia care se află în această situație. Se dorește să fie actualizată starea studenților, fiind cunoscut CNP-ul lor, daca sunt plecați sau au fost plecați în trecut într-un proiect Erasmus în care se vorbește limba învățată de ei, noua stare fiind „Erasmus Acreditat”. De asemenea, studenții din an terminal își păstrează situația academică.

## Rezolvarea

CREATE OR REPLACE FUNCTION erasmus\_acreditat

(vin\_cnp IN NUMBER,

vin\_limba IN VARCHAR2)

RETURN NUMBER

IS

nr\_err number;

ans number;

aux number;

my\_no\_data\_found exception;

no\_students\_to\_change exception;

vals\_modif number := 0;

type tablou is table of student.id\_student%TYPE;

v\_list\_students tablou := tablou();

begin

select count(\*)

into aux

from STUDENT s

where s.cnp = vin\_cnp;

if aux = 0 then

nr\_err := 1;

raise my\_no\_data\_found;

end if;

select count(\*)

into aux

from tari t

where lower(t.limba) = lower(vin\_limba);

if aux = 0 then

nr\_err := 2;

raise my\_no\_data\_found;

end if;

----select \* from erasmus;

select count(\*)

into aux

from ERASMUS e

join studii\_strainatate s2 on e.id\_erasmus = s2.id\_erasmus

join tari t2 on t2.id\_tara = s2.id\_tara

where lower(t2.limba) = lower(vin\_limba) and

e.data\_inceput < sysdate;

if aux = 0 then

nr\_err := 3;

raise my\_no\_data\_found;

end if;

select distinct st.id\_student

bulk collect into v\_list\_students

from studii\_strainatate

join STUDENT st on st.id\_student = studii\_strainatate.id\_student

join tari t3 on t3.id\_tara = studii\_strainatate.id\_tara

join erasmus e2 on studii\_strainatate.id\_erasmus = e2.id\_erasmus

where st.cnp = vin\_cnp and

lower(t3.limba) = lower(vin\_limba) and

e2.data\_inceput < sysdate;

for i in v\_list\_students.first..v\_list\_students.last loop

DBMS\_OUTPUT.PUT\_LINE(v\_list\_students(i));

update student s

set s.tip\_student = 'Erasmus Acreditat'

where (s.an\_studiu < 3) and s.id\_student = v\_list\_students(i);

vals\_modif := vals\_modif + SQL%ROWCOUNT;

end loop;

if vals\_modif = 0 then

raise no\_students\_to\_change;

end if;

return vals\_modif;

exception

WHEN my\_no\_data\_found THEN

if nr\_err = 1 then

DBMS\_OUTPUT.PUT\_LINE('Nu s-a gasit niciun student cu cnp-ul ' || vin\_cnp);

elsif nr\_err = 2 then

DBMS\_OUTPUT.PUT\_LINE('Nu s-a gasit niciun erasmus cu limba ' || vin\_limba);

elsif nr\_err = 3 then

DBMS\_OUTPUT.PUT\_LINE('Nu s-a gasit niciun erasmus cu limba ' || vin\_limba || 'care sa fie inceput');

end if;

RETURN -20001;

WHEN no\_students\_to\_change THEN

DBMS\_OUTPUT.PUT\_LINE('Niciun student nu e in erasmus acreditat');

RETURN -20002;

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Alta eroare');

DBMS\_OUTPUT.PUT\_LINE ('Codul erorii: ' || SQLCODE);

DBMS\_OUTPUT.PUT\_LINE ('Mesajul erorii: ' || SQLERRM);

RETURN -20000;

end erasmus\_acreditat;

/

begin

commit;

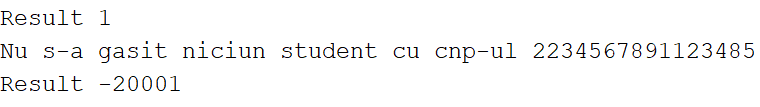
DBMS\_OUTPUT.PUT\_LINE('Result ' || erasmus\_acreditat(1234567891123456, 'Engleza'));

DBMS\_OUTPUT.PUT\_LINE('Result ' || erasmus\_acreditat(2234567891123485, 'Chineza'));

rollback;

end;

/



# **Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să utilizeze într-o singură comandă SQL 5 dintre tabelele definite. Tratați toate excepțiile care pot apărea, incluzând excepțiile NO\_DATA\_FOUND și TOO\_MANY\_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.**

## Cerința

Se cer toți studenții de la buget care au bursa maximă minim K, care au participat la un Erasmus început cu cel mult X ani înainte de azi, cu numele T și moneda din țara respectivă L și au media de admitere minim 9,50.

## Rezolvarea

CREATE OR REPLACE PROCEDURE studenti

(vin\_n in number, ----bursa minima

vin\_data in number,----data minima pt erasmus

vin\_moneda in VARCHAR2, ----moneda tarii

vin\_nume in VARCHAR2) ---numele erasmusului

IS

---select \* from erasmus;

---select \* from caz\_social;

nr\_err number;

ind number := 0;

id\_std buget.id\_student%TYPE;

type tablou is table of buget.id\_student%TYPE;

v\_list\_students tablou := tablou();

type date\_student is record(

id\_stud2 student.id\_student%TYPE,

nume\_student student.nume%TYPE,

prenume\_student student.prenume%TYPE

);

type tablou21 is table of date\_student;

v\_list\_students\_fin tablou21 := tablou21();

type tablou2 is table of date\_student INDEX BY PLS\_INTEGER;

v\_list\_students\_aux tablou2;

type tablou3 is table of erasmus.data\_inceput%TYPE INDEX BY PLS\_INTEGER;

v\_aux\_erasmus\_data tablou3;

type tablou4 is table of erasmus.nume%TYPE INDEX BY PLS\_INTEGER;

v\_aux\_nume\_erasmus tablou4;

negative exception;

my\_no\_data\_found exception;

no\_students\_incase exception;

no\_erasmus\_data exception;

too\_many\_erasmus exception;

BEGIN

if vin\_data < 0 then

raise negative;

end if;

select data\_inceput

bulk collect into v\_aux\_erasmus\_data

from erasmus

where data\_inceput >= (sysdate - vin\_data);

if sql%notfound then

nr\_err := 1;

raise my\_no\_data\_found;

end if;

select nume

bulk collect into v\_aux\_nume\_erasmus

from erasmus

where nume = vin\_nume;

if sql%rowcount > 1 then

raise too\_many\_erasmus;

elsif sql%notfound then

nr\_err := 3;

raise my\_no\_data\_found;

end if;

select bg.id\_student

bulk collect into v\_list\_students

from buget bg

join (

select b.id\_student, MAX(suma) suma\_max

from bursa b

group by b.id\_student

) aux on bg.id\_student = aux.id\_student

where aux.suma\_max >= vin\_n;

if v\_list\_students.count = 0 then

nr\_err := 2;

raise my\_no\_data\_found;

end if;

for i in v\_list\_students.first..v\_list\_students.last loop

id\_std := v\_list\_students(i);

select

st.id\_student,

st.nume,

st.prenume

bulk collect into v\_list\_students\_aux

from student st

where st.id\_student = id\_std and st.id\_student in(

select distinct

R.id\_student

from studii\_strainatate R

join erasmus E2 on R.id\_erasmus = E2.id\_erasmus

join tari t1 on R.id\_tara = t1.id\_tara

join buget bg on bg.id\_student = R.id\_student

where E2.data\_inceput >= (sysdate - vin\_data) and

lower(t1.moneda) = lower(vin\_moneda) and

lower(E2.nume) = lower(vin\_nume) and

bg.medie\_admitere > 9.5

);

if SQL%ROWCOUNT = 1 then

v\_list\_students\_fin.extend();

v\_list\_students\_fin(v\_list\_students\_fin.last) := v\_list\_students\_aux(v\_list\_students\_aux.first);

end if;

end loop;

if v\_list\_students\_fin.count = 0 then

raise no\_students\_incase;

end if;

for i in v\_list\_students\_fin.first..v\_list\_students\_fin.last loop

ind := ind + 1;

DBMS\_OUTPUT.PUT\_LINE('Student ' || ind || ') id: ' || v\_list\_students\_fin(i).id\_stud2 || 'nume: '

|| v\_list\_students\_fin(i).nume\_student || ' ' || v\_list\_students\_fin(i).prenume\_student);

end loop;

EXCEPTION

WHEN my\_no\_data\_found THEN

if nr\_err = 1 then

DBMS\_OUTPUT.PUT\_LINE('Nu exista erasmus dupa data ' || vin\_data);

elsif nr\_err = 2 then

DBMS\_OUTPUT.PUT\_LINE('Nu sunt studenti la buget care au o bursa mai mare de ' || vin\_n);

elsif nr\_err = 3 then

DBMS\_OUTPUT.PUT\_LINE('Nu exista erasmus cu numele ' || vin\_nume);

end if;

RAISE\_APPLICATION\_ERROR(-20001, 'Nu s-au gasit date');

WHEN no\_students\_incase THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista niciun student la buget care sa respecte toate conditiile');

RAISE\_APPLICATION\_ERROR(-20002, 'Nu exista niciun student la buget care sa respecte toate

conditiile');

WHEN too\_many\_erasmus THEN

DBMS\_OUTPUT.PUT\_LINE('Exista mai multe erasmus cu acest nume');

RAISE\_APPLICATION\_ERROR(-20003, 'Exista mai multe erasmus cu acest nume');

WHEN negative THEN

DBMS\_OUTPUT.PUT\_LINE('Numarul de ani trebuie sa fie pozitiv');

RAISE\_APPLICATION\_ERROR(-20004, 'Numarul de ani trebuie sa fie pozitiv');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Alta eroare');

DBMS\_OUTPUT.PUT\_LINE ('Codul erorii: ' || SQLCODE);

DBMS\_OUTPUT.PUT\_LINE ('Mesajul erorii: ' || SQLERRM);

RAISE\_APPLICATION\_ERROR(-20000, 'Alta eroare!');

END studenti;

/

begin

studenti(800, 2, 'EUR', 'Hai la lac!');

--studenti(800000, 1, 'EUR', 'Hai la lac!');

--studenti(800000, 1, 'EUR', 'Hai la nicaieri!');

--studenti(800000, -1, 'EUR', 'Hai la lac!');

/\*Actualizam datele pentru a vedea mai multe afisari\*/

/\*update tari

set moneda = 'AUR'

where id\_tara = 1047;

commit;

studenti(800, 2, 'GBP', 'Hai la lac!');

update tari

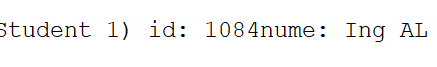
set moneda = 'EUR'

where id\_tara = 1047;

commit;\*/

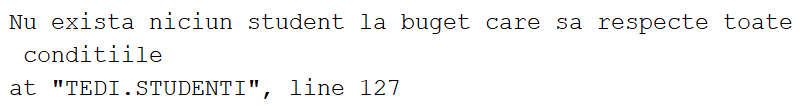
end;

/







# **11. Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul. Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.**

## Cerința

Universitatea a decis ca în momentul în care un student se tranferă la altă universitate, acesta să fie șters din baza de date doar dacă acesta are mai puțin de 2 burse de merit deoarece în această situație este nevoie de diferite date deste student. În alte situații, atât studentul cât și bursele acestuia se șterg din baza de date.

## Rezolvarea

create or replace trigger stergeburse

for delete or update

on student

referencing old as val\_veche

compound trigger

n number := 0;

m number := 0;

TYPE tip\_burse IS RECORD (

id\_bursa bursa.id\_bursa%TYPE,

id\_student bursa.id\_student%TYPE,

nume bursa.nume%TYPE

);

TYPE vect\_p IS TABLE OF tip\_burse INDEX BY PLS\_INTEGER;

list\_burse vect\_p;

TYPE vect\_q IS TABLE OF bursa.id\_bursa%TYPE INDEX BY PLS\_INTEGER;

list\_fin\_burse vect\_q;

no\_burse exception;

before statement is

begin

select b.id\_bursa, b.id\_student, b.nume

bulk collect into list\_burse

from bursa b;

end before statement;

before each row is

begin

DBMS\_OUTPUT.PUT\_LINE('Vreau sa sterg ' || :val\_veche.id\_student);

n := 0;

m := 0;

for i in list\_burse.first..list\_burse.last loop

if :val\_veche.id\_student = list\_burse(i).id\_student then

if lower(list\_burse(i).nume) like 'bursa merit%' then

n := n + 1;

end if;

list\_fin\_burse(m) := list\_burse(i).id\_bursa;

m := m + 1;

end if;

end loop;

if n > 1 then

DBMS\_OUTPUT.PUT\_LINE('Stergere esuata!');

RAISE\_APPLICATION\_ERROR(-20000,'Mai multe burse!');

end if;

end before each row;

after statement is

begin

DBMS\_OUTPUT.PUT\_LINE('S-a sters studentul si cele ' || m || ' burse:');

for i in list\_fin\_burse.first..list\_fin\_burse.last loop

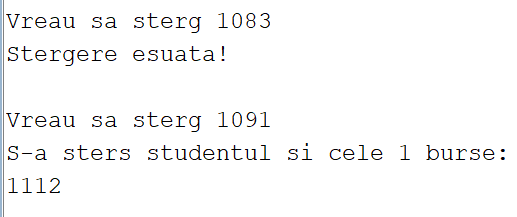
DBMS\_OUTPUT.PUT(list\_fin\_burse(i) || ' ');

end loop;

DBMS\_OUTPUT.PUT\_LINE(' ');

end after statement;

end;



# **Rezolvat împreună cu 10.**

# **Definiți un trigger de tip LDD. Declanșați trigger-ul.**

## Cerința

Când este șters ceva din baza de date, se verifică daca toate FK-urile sunt active. Altfel se va arunca o eroare.

Ca să se șteargă o tabelă din baza de date, prima dată se verifică daca aceasta are toate FK-uri active. Dacă nu, se va arunca o eroare, altfel este permis.

## Rezolvarea

CREATE OR REPLACE TRIGGER boolFK

BEFORE DROP ON DATABASE

DECLARE

vin\_tab VARCHAR2(100);

vin\_cns VARCHAR2(100);

vin\_stt VARCHAR2(100);

BEGIN

FOR tab IN (SELECT table\_name, constraint\_name, status

FROM user\_constraints

WHERE constraint\_type = 'R') LOOP

vin\_tab := tab.table\_name;

vin\_cns := tab.constraint\_name;

vin\_stt := tab.status;

IF vin\_stt <> 'ENABLED' THEN

RAISE\_APPLICATION\_ERROR(-20000, 'Nu avem ' || vin\_cns || ' pe ' || vin\_tab);

END IF;

END LOOP;

END;

/

create table parinte1

(

v varchar2(30)

constraint Ppk

primary key

);

create table copil1

(

vi varchar2(50)

constraint Cpk

primary key,

vii varchar2(30)

);

alter table copil1

add CONSTRAINT Cfk

FOREIGN KEY (vii)

REFERENCES parinte1(v)

ON DELETE SET NULL;

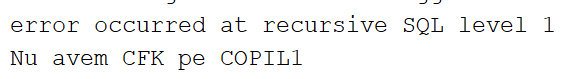
alter table copil1

enable constraint Cfk;

drop table copil1;

rollback;





# **Definiți un pachet care să conțină toate obiectele definite în cadrul proiectului.**

## Rezolvarea

create or replace package studenti\_pachet2 is

PROCEDURE findMostNActiveStudent

(v\_nrcomp NUMBER DEFAULT 1);

PROCEDURE proc1

(v\_in NUMBER DEFAULT 0);

FUNCTION erasmus\_acreditat

(vin\_cnp IN NUMBER,

vin\_limba IN VARCHAR2)

RETURN NUMBER;

PROCEDURE studenti

(vin\_n in number,

vin\_data in number,

vin\_moneda in varchar2,

vin\_nume in varchar2);

end studenti\_pachet2;

create or replace package body studenti\_pachet2 is

PROCEDURE findMostNActiveStudent

(v\_nrcomp NUMBER DEFAULT 1)

IS

/\*Record\*/

type student\_info is record (

id\_student STUDENT.id\_student%TYPE,

nume\_student STUDENT.nume%TYPE,

id\_bursa BURSA.id\_bursa%TYPE

);

/\*Tablou imbricat\*/

type tablou\_imbricat is table of student\_info;

v\_list\_students tablou\_imbricat := tablou\_imbricat();

v\_old\_student STUDENT.id\_student%TYPE;

v\_nume STUDENT.nume%TYPE;

/\*Tablou indexat\*/

type tablou\_indexat is table of bursa.id\_bursa%TYPE index by pls\_integer;

v\_list\_burse tablou\_indexat;

v\_allburse number;

/\*Vector\*/

type vector is varray(150) of STUDENT.id\_student%TYPE;

v\_list\_studentids vector := vector();

i number;

j number;

minim number;

contor number;

too\_many\_burse exception;

too\_many\_burse\_cautate exception;

BEGIN

select count(\*)

into v\_allburse

from bursa;

if(v\_nrcomp > v\_allburse) then

raise too\_many\_burse;

end if;

select BAUX.ID\_BURSA

bulk collect into v\_list\_burse

from BURSA BAUX

join (select B.id\_bursa, count(\*) as nr\_burse\_sociale

from BURSA B

join caz\_social C2 on C2.id\_bursa = B.id\_bursa

where lower(C2.nr\_adeverinte) > 2

GROUP BY B.id\_bursa) D on D.id\_bursa = BAUX.id\_bursa

order by D.nr\_burse\_sociale desc, BAUX.suma;

if(v\_nrcomp > v\_list\_burse.COUNT) then

raise too\_many\_burse\_cautate;

end if;

select s2.id\_student, s2.nume, b.id\_bursa

bulk collect into v\_list\_students

from bursa b

join student s2 on s2.id\_student = b.id\_student

order by s2.id\_student;

v\_old\_student := v\_list\_students(1).id\_student;

contor := 0;

minim := v\_list\_students.count;

for i in v\_list\_students.first..v\_list\_students.last loop

if(v\_old\_student <> v\_list\_students(i).id\_student) then

if(minim = contor and contor <> 0) then

v\_list\_studentids.extend;

v\_list\_studentids(v\_list\_studentids.last) := v\_old\_student;

elsif(minim > contor and contor <> 0) then

v\_list\_studentids.delete;

v\_list\_studentids.extend;

v\_list\_studentids(v\_list\_studentids.last) := v\_old\_student;

minim := contor;

end if;

v\_old\_student := v\_list\_students(i).id\_student;

contor:= 0;

end if;

for j in 1..v\_nrcomp loop

if(v\_list\_students(i).id\_bursa = v\_list\_burse(j)) then

contor := contor + 1;

end if;

end loop;

end loop;

if(minim = contor and contor <> 0) then

v\_list\_studentids.extend;

v\_list\_studentids(v\_list\_studentids.last) := v\_old\_student;

elsif(minim > contor and contor <> 0) then

v\_list\_studentids.delete;

v\_list\_studentids.extend;

v\_list\_studentids(v\_list\_studentids.last) := v\_old\_student;

minim := contor;

end if;

for i in v\_list\_studentids.first..v\_list\_studentids.last loop

select nume

into v\_nume

from student

where id\_student = v\_list\_studentids(i);

if(v\_nume is not null) then

DBMS\_OUTPUT.PUT\_LINE('Student ' || v\_nrcomp || '-activ: id = ' || v\_list\_studentids(i) || ' nume = '

|| v\_nume);

else

DBMS\_OUTPUT.PUT\_LINE('Student ' || v\_nrcomp || '-activ: id = ' || v\_list\_studentids(i) || ' fara

nume');

end if;

end loop;

EXCEPTION

when too\_many\_burse then

RAISE\_APPLICATION\_ERROR(-20001,'Prea putine burse cerute');

when too\_many\_burse\_cautate then

RAISE\_APPLICATION\_ERROR(-20002,'Prea multe acte cerute');

when TOO\_MANY\_ROWS then

RAISE\_APPLICATION\_ERROR(-20003,'Prea multe burse gasite cu acelasi id');

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20004,'Nu s-au gasit datele');

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20000,'Alta eroare!');

END findMostNActiveStudent;

PROCEDURE proc1

(v\_in NUMBER DEFAULT 0)

IS

contor number;

flag number := 0;

cursor c is

select s.id\_student, count(\*) nr\_burse

from STUDENT s

join bursa B2 on s.id\_student = B2.id\_student

group by s.id\_student

having count(\*) <= v\_in;

cursor u(v\_idstudent STUDENT.id\_student%type) is

select \*

from bursa

where id\_student = v\_idstudent and id\_student in (

select id\_student

from studii\_strainatate

join ERASMUS E3 on E3.id\_erasmus = studii\_strainatate.id\_erasmus

where E3.durata >= 100

)

for update of suma nowait ;

no\_students exception;

BEGIN

for student in c loop

flag := 1;

contor := 0;

for j in u(student.id\_student) loop

if(lower(j.nume) like 'bursa merit%') then

contor := contor + 1;

update BURSA

set suma = suma\*1.2

where current of u;

end if;

end loop;

DBMS\_OUTPUT.PUT('Studentul '|| student.id\_student);

if(contor = 0) then

DBMS\_OUTPUT.PUT\_LINE(' nu si-a marit burse de merit');

else

DBMS\_OUTPUT.PUT\_LINE(' si-a marit '|| contor || ' burse');

end if;

rollback; -------------------------------

end loop;

if(flag = 0) then

raise no\_students;

end if;

EXCEPTION

when INVALID\_CURSOR then

RAISE\_APPLICATION\_ERROR(-20001, 'Cursorul este inchis');

when CURSOR\_ALREADY\_OPEN then

RAISE\_APPLICATION\_ERROR(-20002, 'Cursor deja deschis');

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20003,'Nu s-au gasit date');

WHEN no\_students THEN

RAISE\_APPLICATION\_ERROR(-20004,'Nu sunt studenti cu acest nr minim de burse');

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20000,'Alta eroare!');

END proc1;

FUNCTION erasmus\_acreditat

(vin\_cnp IN NUMBER,

vin\_limba IN VARCHAR2)

RETURN NUMBER

IS

nr\_err number;

ans number;

aux number;

my\_no\_data\_found exception;

no\_students\_to\_change exception;

vals\_modif number := 0;

type tablou is table of student.id\_student%TYPE;

v\_list\_students tablou := tablou();

begin

select count(\*)

into aux

from STUDENT s

where s.cnp = vin\_cnp;

if aux = 0 then

nr\_err := 1;

raise my\_no\_data\_found;

end if;

select count(\*)

into aux

from tari t

where lower(t.limba) = lower(vin\_limba);

if aux = 0 then

nr\_err := 2;

raise my\_no\_data\_found;

end if;

----select \* from erasmus;

select count(\*)

into aux

from ERASMUS e

join studii\_strainatate s2 on e.id\_erasmus = s2.id\_erasmus

join tari t2 on t2.id\_tara = s2.id\_tara

where lower(t2.limba) = lower(vin\_limba) and

e.data\_inceput < sysdate;

if aux = 0 then

nr\_err := 3;

raise my\_no\_data\_found;

end if;

select distinct st.id\_student

bulk collect into v\_list\_students

from studii\_strainatate

join STUDENT st on st.id\_student = studii\_strainatate.id\_student

join tari t3 on t3.id\_tara = studii\_strainatate.id\_tara

join erasmus e2 on studii\_strainatate.id\_erasmus = e2.id\_erasmus

where st.cnp = vin\_cnp and

lower(t3.limba) = lower(vin\_limba) and

e2.data\_inceput < sysdate;

for i in v\_list\_students.first..v\_list\_students.last loop

DBMS\_OUTPUT.PUT\_LINE(v\_list\_students(i));

update student s

set s.tip\_student = 'Erasmus Acreditat'

where (s.an\_studiu < 3) and s.id\_student = v\_list\_students(i);

vals\_modif := vals\_modif + SQL%ROWCOUNT;

end loop;

if vals\_modif = 0 then

raise no\_students\_to\_change;

end if;

return vals\_modif;

exception

WHEN my\_no\_data\_found THEN

if nr\_err = 1 then

DBMS\_OUTPUT.PUT\_LINE('Nu s-a gasit niciun student cu cnp-ul ' || vin\_cnp);

elsif nr\_err = 2 then

DBMS\_OUTPUT.PUT\_LINE('Nu s-a gasit niciun erasmus cu limba ' || vin\_limba);

elsif nr\_err = 3 then

DBMS\_OUTPUT.PUT\_LINE('Nu s-a gasit niciun erasmus cu limba ' || vin\_limba || 'care sa fie inceput');

end if;

RETURN -20001;

WHEN no\_students\_to\_change THEN

DBMS\_OUTPUT.PUT\_LINE('Niciun student nu e in erasmus acreditat');

RETURN -20002;

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Alta eroare');

DBMS\_OUTPUT.PUT\_LINE ('Codul erorii: ' || SQLCODE);

DBMS\_OUTPUT.PUT\_LINE ('Mesajul erorii: ' || SQLERRM);

RETURN -20000;

end erasmus\_acreditat;

PROCEDURE studenti

(vin\_n in number, ----bursa minima

vin\_data in number,----data minima pt erasmus

vin\_moneda in VARCHAR2, ----moneda tarii

vin\_nume in VARCHAR2) ---numele erasmusului

IS

---select \* from erasmus;

---select \* from caz\_social;

nr\_err number;

ind number := 0;

id\_std buget.id\_student%TYPE;

type tablou is table of buget.id\_student%TYPE;

v\_list\_students tablou := tablou();

type date\_student is record(

id\_stud2 student.id\_student%TYPE,

nume\_student student.nume%TYPE,

prenume\_student student.prenume%TYPE

);

type tablou21 is table of date\_student;

v\_list\_students\_fin tablou21 := tablou21();

type tablou2 is table of date\_student INDEX BY PLS\_INTEGER;

v\_list\_students\_aux tablou2;

type tablou3 is table of erasmus.data\_inceput%TYPE INDEX BY PLS\_INTEGER;

v\_aux\_erasmus\_data tablou3;

type tablou4 is table of erasmus.nume%TYPE INDEX BY PLS\_INTEGER;

v\_aux\_nume\_erasmus tablou4;

negative exception;

my\_no\_data\_found exception;

no\_students\_incase exception;

no\_erasmus\_data exception;

too\_many\_erasmus exception;

BEGIN

if vin\_data < 0 then

raise negative;

end if;

select data\_inceput

bulk collect into v\_aux\_erasmus\_data

from erasmus

where data\_inceput >= (sysdate - vin\_data);

if sql%notfound then

nr\_err := 1;

raise my\_no\_data\_found;

end if;

select nume

bulk collect into v\_aux\_nume\_erasmus

from erasmus

where nume = vin\_nume;

if sql%rowcount > 1 then

raise too\_many\_erasmus;

elsif sql%notfound then

nr\_err := 3;

raise my\_no\_data\_found;

end if;

select bg.id\_student

bulk collect into v\_list\_students

from buget bg

join (

select b.id\_student, MAX(suma) suma\_max

from bursa b

group by b.id\_student

) aux on bg.id\_student = aux.id\_student

where aux.suma\_max >= vin\_n;

if v\_list\_students.count = 0 then

nr\_err := 2;

raise my\_no\_data\_found;

end if;

for i in v\_list\_students.first..v\_list\_students.last loop

id\_std := v\_list\_students(i);

select

st.id\_student,

st.nume,

st.prenume

bulk collect into v\_list\_students\_aux

from student st

where st.id\_student = id\_std and st.id\_student in(

select distinct

R.id\_student

from studii\_strainatate R

join erasmus E2 on R.id\_erasmus = E2.id\_erasmus

join tari t1 on R.id\_tara = t1.id\_tara

where E2.data\_inceput >= (sysdate - vin\_data) and

lower(t1.moneda) = lower(vin\_moneda) and

lower(E2.nume) = lower(vin\_nume)

);

if SQL%ROWCOUNT = 1 then

v\_list\_students\_fin.extend();

v\_list\_students\_fin(v\_list\_students\_fin.last) := v\_list\_students\_aux(v\_list\_students\_aux.first);

end if;

end loop;

if v\_list\_students\_fin.count = 0 then

raise no\_students\_incase;

end if;

for i in v\_list\_students\_fin.first..v\_list\_students\_fin.last loop

ind := ind + 1;

DBMS\_OUTPUT.PUT\_LINE('Student ' || ind || ') id: ' || v\_list\_students\_fin(i).id\_stud2 || 'nume: '

|| v\_list\_students\_fin(i).nume\_student || ' ' || v\_list\_students\_fin(i).prenume\_student);

end loop;

EXCEPTION

WHEN my\_no\_data\_found THEN

if nr\_err = 1 then

DBMS\_OUTPUT.PUT\_LINE('Nu exista erasmus dupa data ' || vin\_data);

elsif nr\_err = 2 then

DBMS\_OUTPUT.PUT\_LINE('Nu sunt studenti la buget care au o bursa mai mare de ' || vin\_n);

elsif nr\_err = 3 then

DBMS\_OUTPUT.PUT\_LINE('Nu exista erasmus cu numele ' || vin\_nume);

end if;

RAISE\_APPLICATION\_ERROR(-20001, 'Nu s-au gasit date');

WHEN no\_students\_incase THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista niciun student la buget care sa respecte toate conditiile');

RAISE\_APPLICATION\_ERROR(-20002, 'Nu exista niciun student la buget care sa respecte toate

conditiile');

WHEN too\_many\_erasmus THEN

DBMS\_OUTPUT.PUT\_LINE('Exista mai multe erasmus cu acest nume');

RAISE\_APPLICATION\_ERROR(-20003, 'Exista mai multe erasmus cu acest nume');

WHEN negative THEN

DBMS\_OUTPUT.PUT\_LINE('Numarul de ani trebuie sa fie pozitiv');

RAISE\_APPLICATION\_ERROR(-20004, 'Numarul de ani trebuie sa fie pozitiv');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Alta eroare');

DBMS\_OUTPUT.PUT\_LINE ('Codul erorii: ' || SQLCODE);

DBMS\_OUTPUT.PUT\_LINE ('Mesajul erorii: ' || SQLERRM);

RAISE\_APPLICATION\_ERROR(-20000, 'Alta eroare!');

END studenti;

end studenti\_pachet2;

/

declare

v\_x number := &p;

begin

studenti\_pachet2.findMostNActiveStudent(2);

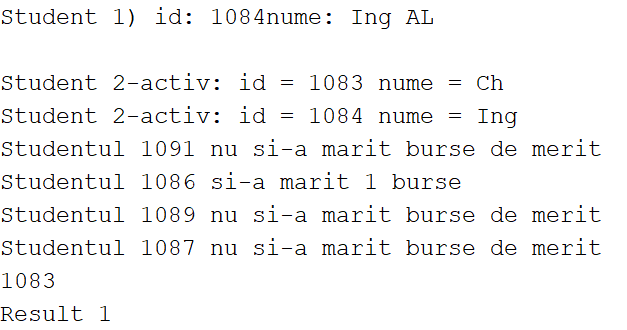
studenti\_pachet2.proc1(v\_x);

DBMS\_OUTPUT.PUT\_LINE('Result ' || studenti\_pachet2.erasmus\_acreditat(1234567891123456, 'Engleza'));

studenti\_pachet2.studenti(800, 2, 'EUR', 'Hai la lac!');

rollback;

end;



# **Definiți un pachet care să includă tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date definite (minim 2 tipuri de date, minim 2 funcții, minim 2 proceduri).**

## Cerința

Universitatea vrea să premieze studenții. Studenții care au cel puțin două burse, primesc gratuitate la un Proiect Erasmus. Pentru asta este nevoie de un pachet care să verifice daca un student anume are măcar două burse și dacă da, să i se ofere un Erasmus. De asemenea, trebuie verificat dacă studentul are deja Erasmusul în plan, iar dacă da, se afișează un mesaj sugestiv. Pentru un calcul de buget, Universitatea are nevoie și de bursa maximă a studentului. În plus, dacă studentul are media de admitere de minim 9.50, acesta primește o bursă Erasmus în valoare de bursa maximă a sa, cu titlul „Bursă Erasmus”.

## Rezolvarea

create or replace package toerasmus is

PROCEDURE init;

PROCEDURE add\_erasmus

(v\_id\_erasmus IN studii\_strainatate.id\_erasmus%TYPE,

v\_id\_student IN studii\_strainatate.id\_student%TYPE);

FUNCTION adaugare\_valida

(v\_id\_student IN studii\_strainatate.id\_student%TYPE,

v\_id\_bursa\_max OUT bursa.id\_bursa%TYPE)

RETURN BOOLEAN;

FUNCTION verif\_medie

(vv\_id\_student IN student.id\_student%TYPE)

RETURN BOOLEAN;

end toerasmus;

/

create or replace package body toerasmus is

initializare boolean := false;

type bursa\_data is record(

id\_student bursa.id\_bursa%TYPE,

id\_bursa bursa.id\_student%TYPE,

nr\_burse number

);

type tablou1 is table of bursa\_data;

type tablou11 is table of tablou1;

v\_init\_students tablou11 := tablou11();

PROCEDURE init IS

v\_burse tablou1 := tablou1();

id\_std number;

BEGIN

for j in (select id\_student from student) loop

id\_std := j.id\_student;

select id\_student, id\_bursa, MAX(suma) max\_brs

bulk collect into v\_burse

from bursa

group by id\_student, id\_bursa

having id\_student = id\_std

order by max\_brs desc, id\_bursa;

v\_init\_students.extend();

v\_init\_students(v\_init\_students.last) := v\_burse;

end loop;

initializare := true;

END init;

FUNCTION verif\_medie

(vv\_id\_student IN student.id\_student%TYPE)

RETURN BOOLEAN is

v\_medie number;

begin

select medie\_admitere

into v\_medie

from buget

where id\_student = vv\_id\_student;

if v\_medie > 9.5 then

return true;

return false;

end if;

end verif\_medie;

PROCEDURE add\_erasmus

(v\_id\_erasmus IN studii\_strainatate.id\_erasmus%TYPE,

v\_id\_student IN studii\_strainatate.id\_student%TYPE) IS

nr\_err number;

aux\_student\_id student.id\_student%TYPE;

aux\_erasmus\_id erasmus.id\_erasmus%TYPE;

type tablou2 is table of studii\_strainatate.id\_relatie\_ss%TYPE;

v\_echs tablou2 := tablou2();

last\_id number;

id\_bursa\_max bursa.id\_bursa%TYPE;

erasmus\_existent exception;

venit\_insuficient exception;

fara\_initializare exception;

burse\_insuficiente exception;

BEGIN

initializare := false;

init();

if not initializare then

raise fara\_initializare;

end if;

nr\_err := 1;

select id\_student

into aux\_student\_id

from student

where id\_student = v\_id\_student;

nr\_err := 2;

select id\_erasmus

into aux\_erasmus\_id

from erasmus

where id\_erasmus = v\_id\_erasmus;

select id\_relatie\_ss

bulk collect into v\_echs

from studii\_strainatate

where ID\_erasmus = v\_id\_erasmus and id\_student = v\_id\_student;

if SQL%ROWCOUNT >= 1 THEN

raise erasmus\_existent;

end if;

if not adaugare\_valida(v\_id\_student, id\_bursa\_max) then

raise burse\_insuficiente;

end if;

select nvl(max(id\_relatie\_ss), 0)

into last\_id

from studii\_strainatate r;

---DBMS\_OUTPUT.PUT\_LINE(v\_id\_student);

---DBMS\_OUTPUT.PUT\_LINE(v\_id\_erasmus);

---insert into studii\_strainatate values (1105, 1083, 1044 , 1052);

insert into studii\_strainatate values (last\_id + 1, v\_id\_student, 1044 , v\_id\_erasmus);

if verif\_medie(v\_id\_student) then

insert into bursa values(last\_id + 1, 'Bursa Erasmus', (select suma from bursa where id\_bursa = id\_bursa\_max), v\_id\_student);

end if;

---baga\_insert(v\_id\_erasmus, v\_id\_student);

DBMS\_OUTPUT.PUT\_LINE('Student adaugat la erasmus');

DBMS\_OUTPUT.PUT\_LINE('Cea mai mare bursa este ' || id\_bursa\_max);

---afisare\_detalii\_bursa(id\_bursa\_max);

rollback;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

if nr\_err = 1 then

DBMS\_OUTPUT.PUT\_LINE('Nu exista student cu id = ' || v\_id\_student);

elsif nr\_err = 2 then

DBMS\_OUTPUT.PUT\_LINE('Nu exista erasmus cu id = ' || v\_id\_erasmus);

end if;

RAISE\_APPLICATION\_ERROR(-20001, 'Nu s-au gasit date');

WHEN erasmus\_existent THEN

DBMS\_OUTPUT.PUT\_LINE('Studentul ' || v\_id\_student || ' are deja erasmusul ' ||

v\_id\_erasmus);

RAISE\_APPLICATION\_ERROR(-20002, 'Studentul are acest erasmus!');

WHEN fara\_initializare THEN

DBMS\_OUTPUT.PUT\_LINE('A aparut o eroare la initializare');

RAISE\_APPLICATION\_ERROR(-20003, 'Eroare la initializare!');

WHEN burse\_insuficiente THEN

DBMS\_OUTPUT.PUT\_LINE('Studentul nu are mai mult de o bursa');

RAISE\_APPLICATION\_ERROR(-20004, 'Burse insuficiente!');

END add\_erasmus;

FUNCTION adaugare\_valida

(v\_id\_student IN studii\_strainatate.id\_student%TYPE,

v\_id\_bursa\_max OUT bursa.id\_bursa%TYPE)

RETURN BOOLEAN IS

v\_nr\_burse number;

flag boolean := true;

poz number;

no\_burse exception;

BEGIN

for i in v\_init\_students.first..v\_init\_students.last loop

if v\_init\_students(i)(v\_init\_students(i).first).id\_student = v\_id\_student then

v\_nr\_burse := v\_init\_students(i).count;

flag := false;

poz := i;

exit;

end if;

end loop;

if flag then

return false;

end if;

if v\_nr\_burse < 2 then

return false;

end if;

v\_id\_bursa\_max := v\_init\_students(poz)(v\_init\_students(poz).FIRST).id\_bursa;

return true;

EXCEPTION

WHEN no\_burse THEN

DBMS\_OUTPUT.PUT\_LINE('Studentul ' || v\_id\_student || ' nu are nicio bursa.');

return false;

END adaugare\_valida;

end toerasmus;

/

rollback;

begin

toerasmus.add\_erasmus(1051, 1083);

---toerasmus.add\_erasmus(1051, 1085);

---toerasmus.add\_erasmus(1, 1083);

---toerasmus.add\_erasmus(1050, 1083);

end;

/

