**Proiect SGBD**

**Baza de date pentru un liceu**

1. Prezentati pe scurt baza de date (utilitatea ei).

Modelul de date va gestiona informatii legate de organizarea administrativa a unui liceu. Acesta va cuprinde:

-cateva biblioteci ce contin documente/carti care pot constitui material didactic;

-cateva cabinete medicale dotate cu echipamentul necesar ingrijirii, la nevoie, a elevilor si a cadrelor didactice;

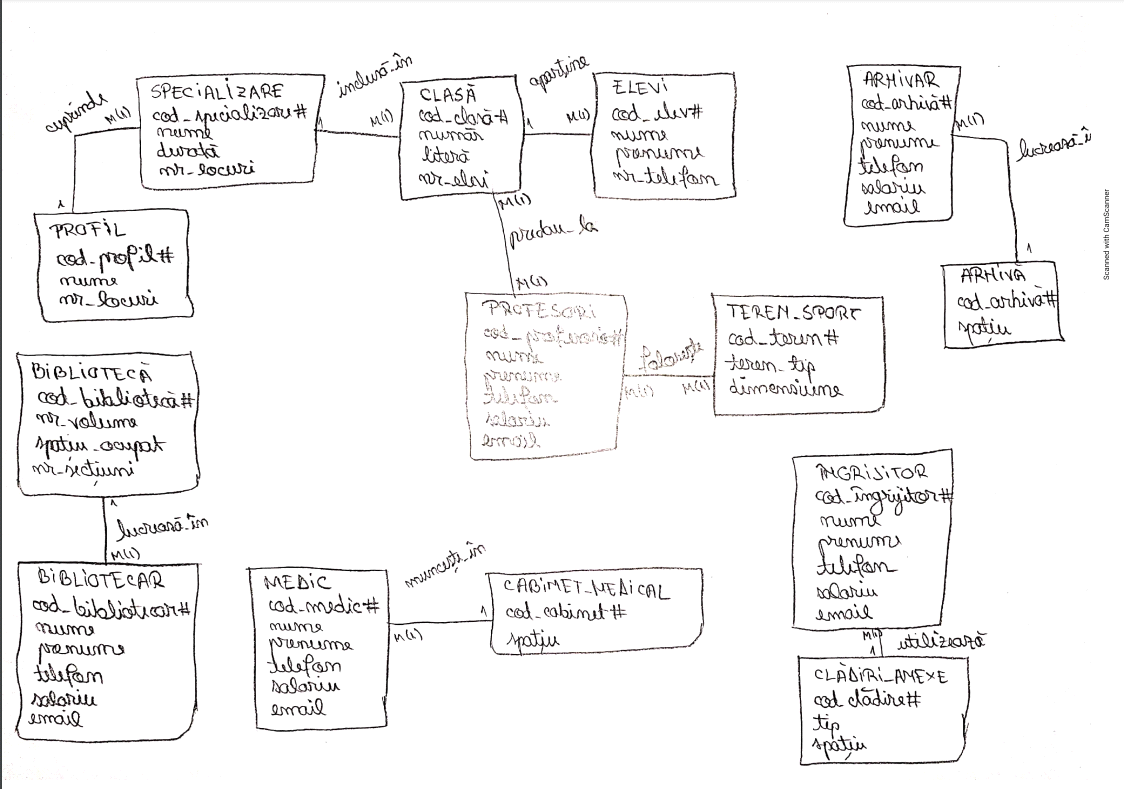
-cateva cladiri anexe in care ingrijitorii isi depoziteaza ustensilele necesare muncii prestate;

-cateva arhive in care sunt pastrate toate documentele importante ale liceului, cum ar fi cataloagele;

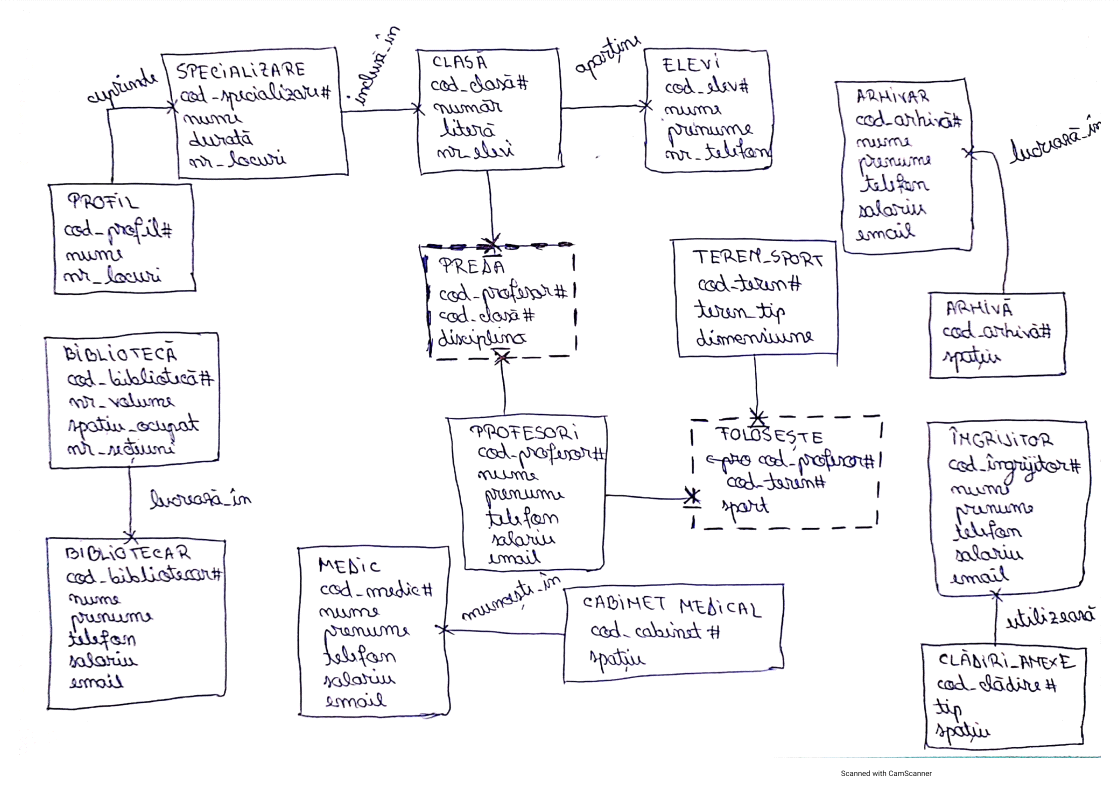
-gestiunea profilurilor si specializarilor la care elevii pot studia;

-gestiunea claselor care apartin specializarilor, a elevilor care apartin claselor;

1. Realizati **diagrama entitate-relatie** (ERD).



1. Pornind de la diagrama entitate-relatie realizati **diagrama conceptuala** a modelului propus, integrand toate atributele necesare.



1. Implementati in Oracle diagrama conceptuala realizata: definiti toate tabelele, implementand toate constrangerile de integritate necesare (chei primare, chei externe etc).

Tabelul „PROFIL”:

create table PROFIL(PROFIL\_ID number(4) primary key,

PROFIL\_NUME varchar2(40),

NR\_LOCURI number(2));

Tabelul „SPECIALIZARE”:

create table SPECIALIZARE (SPECIALIZARE\_ID number(4) primary key,

SPECIALIZARE\_NUME varchar2(40),

SPECIALIZARE\_DURATA number(1),

SPECIALIZARE\_NR\_LOCURI number(2));

ALTER TABLE SPECIALIZARE

ADD PROFIL\_ID NUMBER(4);

ALTER TABLE SPECIALIZARE

ADD FOREIGN KEY (PROFIL\_ID) REFERENCES PROFIL(PROFIL\_ID);

Tabelul „CLASA”:

create table CLASA (CLASA\_ID number(4) primary key,

CLASA\_NUMAR number(2),

CLASA\_LITERA varchar2(1),

NR\_ELEVI number(2));

ALTER TABLE CLASA

ADD SPECIALIZARE\_ID NUMBER(4);

ALTER TABLE CLASA

ADD FOREIGN KEY (SPECIALIZARE\_ID) REFERENCES SPECIALIZARE(SPECIALIZARE\_ID);

Tabelul „ELEVI”:

create table ELEVI (ELEV\_ID number(4) primary key,

ELEV\_NUME varchar2(30),

ELEV\_PRENUME varchar2(30),

NR\_TELEFON number(10));

ALTER TABLE ELEVI

ADD CLASA\_ID NUMBER(4);

ALTER TABLE ELEVI

ADD FOREIGN KEY (CLASA\_ID) REFERENCES CLASA(CLASA\_ID);

Tabelul „BIBLIOTECA”:

create table BIBLIOTECA (BIBLIOTECA\_ID number(4) primary key,

NR\_VOLUME number(3),

SPATIU\_OCUPAT number(2),

NR\_SECTIUNI number(2));

Tabelul „TEREN SPORT”:

create table TEREN\_SPORT (TEREN\_ID number(4) primary key,

TEREN\_TIP varchar2(30),

DIMENSIUNE number(3));

Tabelul „CLADIRI ANEXE”:

create table CLADIRI\_ANEXE (CLADIRE\_ID number(4) primary key,

CLADIRE\_TIP varchar2(30),

CLADIRE\_SPATIU number(2));

Tabelul „ARHIVA”:

create table ARHIVA (ARHIVA\_ID number(4) primary key,

ARHIVA\_SPATIU number(2));

Tabelul „CABINET MEDICAL”:

create table CABINET\_MEDICAL (CABINET\_ID number(4) primary key,

CABINET\_SPATIU number(2));

Tabelul „PROFESORI”:

create table PROFESORI (PROFESOR\_ID number(4) primary key,

PROFESOR\_NUME varchar2(30),

PROFESOR\_PRENUME varchar2(30),

TELEFON NUMBER(10),

SALARIU number(4),

EMAIL varchar2(30));

Tabelul „INGRIJITOR”:

create table INGRIJITOR (INGRIJITOR\_ID number(4) primary key,

INGRIJITOR\_NUME varchar2(30),

INGRIJITOR\_PRENUME varchar2(30),

I\_TELEFON NUMBER(10),

I\_SALARIU number(4),

I\_EMAIL varchar2(30));

ALTER TABLE INGRIJITOR

ADD CLADIRE\_ID NUMBER(4);

ALTER TABLE INGRIJITOR

ADD FOREIGN KEY (CLADIRE\_ID) REFERENCES CLADIRI\_ANEXE(CLADIRE\_ID);

Tabelul „BIBLIOTECAR”:

create table BIBLIOTECAR (BIBLIOTECAR\_ID number(4) primary key,

BIBLIOTECAR\_NUME varchar2(30),

BIBLIOTECAR\_PRENUME varchar2(30),

B\_TELEFON NUMBER(10),

B\_SALARIU number(4),

B\_EMAIL varchar2(30));

ALTER TABLE BIBLIOTECAR

ADD BIBLIOTECA\_ID NUMBER(4);

ALTER TABLE BIBLIOTECAR

ADD FOREIGN KEY (BIBLIOTECA\_ID) REFERENCES BIBLIOTECA(BIBLIOTECA\_ID);

Tabelul „ARHIVAR”:

create table ARHIVAR (ARHIVAR\_ID number(4) primary key,

ARHIVAR\_NUME varchar2(30),

ARHIVAR\_PRENUME varchar2(30),

A\_TELEFON NUMBER(10),

A\_SALARIU number(4),

A\_EMAIL varchar2(30));

ALTER TABLE ARHIVAR

ADD ARHIVA\_ID NUMBER(4);

ALTER TABLE ARHIVAR

ADD FOREIGN KEY (ARHIVA\_ID) REFERENCES ARHIVA(ARHIVA\_ID);

Tabelul „MEDIC”:

create table MEDIC (MEDIC\_ID number(4) primary key,

MEDIC\_NUME varchar2(30),

MEDIC\_PRENUME varchar2(30),

M\_TELEFON NUMBER(10),

M\_SALARIU number(4),

M\_EMAIL varchar2(30));

ALTER TABLE MEDIC

ADD CABINET\_ID NUMBER(4);

ALTER TABLE MEDIC

ADD FOREIGN KEY(CABINET\_ID) REFERENCES CABINET\_MEDICAL(CABINET\_ID);

Tabelul asociativ „PREDA”:

CREATE TABLE PREDA(PROFESOR\_ID NUMBER(4) REFERENCES PROFESORI(PROFESOR\_ID),

CLASA\_ID NUMBER(4) REFERENCES CLASA(CLASA\_ID),

DISCIPLINA VARCHAR2(30),

PRIMARY KEY(PROFESOR\_ID, CLASA\_ID));

Tabelul asociativ „FOLOSESTE”:

CREATE TABLE FOLOSESTE(PROFESOR\_ID NUMBER(4) REFERENCES PROFESORI(PROFESOR\_ID),

TEREN\_ID NUMBER(4) REFERENCES TEREN\_SPORT(TEREN\_ID),

SPORT VARCHAR2(30),

PRIMARY KEY(PROFESOR\_ID, TEREN\_ID));

1. Adaugati informatii coerente in tabelele create (minim 5 inregistrari pentru fiecare entitate independenta; minim 10 inregistrari pentru tabela asociativa).

Inserari in tabela „PROFIL”:

INSERT INTO PROFIL

VALUES (1, 'REAL', 50);

INSERT INTO PROFIL

VALUES(2, 'UMAN', 40);

INSERT INTO PROFIL

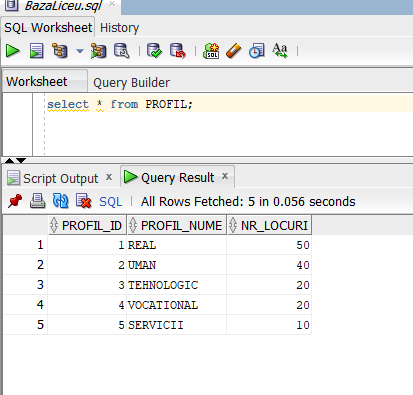
VALUES(3, 'TEHNOLOGIC', 20);

INSERT INTO PROFIL

VALUES(4, 'VOCATIONAL', 20);

INSERT INTO PROFIL

VALUES(5, 'SERVICII', 10);



Inserari in tabela „SPECIALIZARE”:

INSERT INTO SPECIALIZARE

VALUES (6, 'MATEMATICA-INFORMATICA', 4, 40, 1);

INSERT INTO SPECIALIZARE

VALUES (7, 'STIINTELE-NATURII', 4, 28, 1);

INSERT INTO SPECIALIZARE

VALUES (9, 'STIINTE-SOCIALE', 4, 28, 2);

INSERT INTO SPECIALIZARE

VALUES (10, 'MECANICA', 3, 28, 3);

INSERT INTO SPECIALIZARE

VALUES (11, 'ELECTRICA', 3, 28, 3);

INSERT INTO SPECIALIZARE

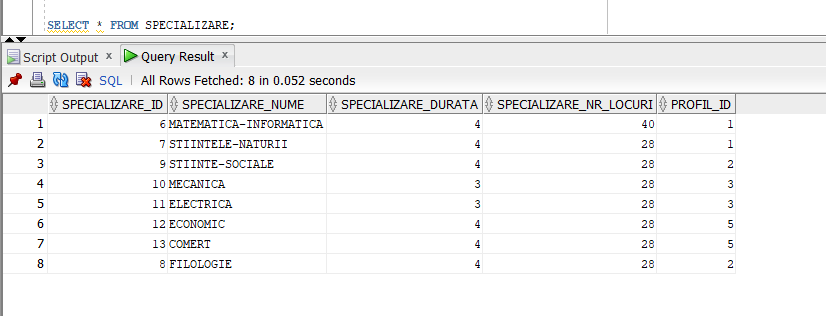
VALUES (12, 'ECONOMIC', 4, 28, 5);

INSERT INTO SPECIALIZARE

VALUES (13, 'COMERT', 4, 28, 5);

INSERT INTO SPECIALIZARE

VALUES (8, 'FILOLOGIE', 4, 28, 2);



Inserari in tabela „CLASA”:

INSERT INTO CLASA

VALUES (1, 9, 'A', 27, 6);

INSERT INTO CLASA

VALUES(2, 10, 'B', 28, 7);

INSERT INTO CLASA

VALUES(3, 11, 'A', 25, 8);

INSERT INTO CLASA

VALUES(4, 9, 'B', 26, 9);

INSERT INTO CLASA

VALUES(5, 12, 'A', 26, 10);

INSERT INTO CLASA

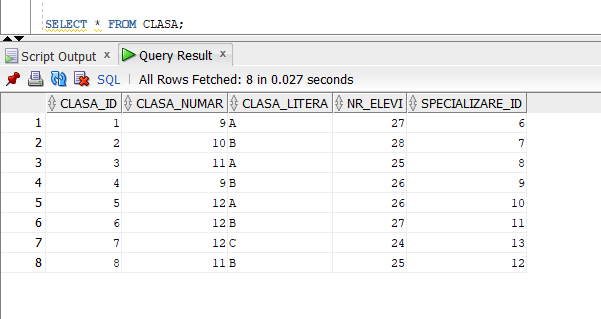
VALUES(6, 12, 'B', 27, 11);

INSERT INTO CLASA

VALUES(7, 12, 'C', 24, 13);

INSERT INTO CLASA

VALUES(8, 11, 'B', 25, 12);



Inserari in tabela „ELEVI”:

INSERT INTO ELEVI

VALUES(10, 'POPESCU', 'ANDREI', 0712345678, 1);

INSERT INTO ELEVI

VALUES(11, 'ANTONESCU', 'DARIA', 0752345678, 1);

INSERT INTO ELEVI

VALUES(12, 'OPREA', 'ANDREEA', 0752545678, 2);

INSERT INTO ELEVI

VALUES(13, 'OPRESCU', 'DANIEL', 0752545978, 2);

INSERT INTO ELEVI

VALUES(14, 'SERBAN', 'MIHAI', 0752845978, 3);

INSERT INTO ELEVI

VALUES(15, 'STANCIU', 'DAVID', 0792845978, 3);

INSERT INTO ELEVI

VALUES(16, 'IONESCU', 'ALINA', 0790845978, 4);

INSERT INTO ELEVI

VALUES(17, 'ION', 'VALENTIN', 0790845078, 5);

INSERT INTO ELEVI

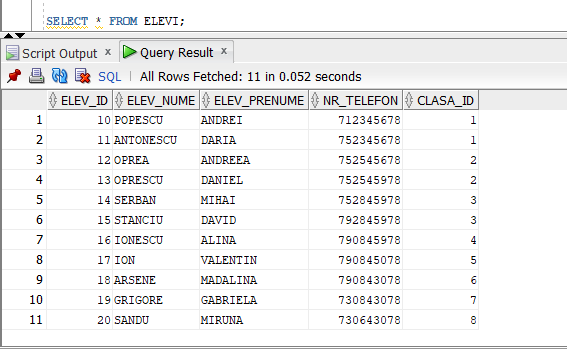
VALUES(18, 'ARSENE', 'MADALINA', 0790843078, 6);

INSERT INTO ELEVI

VALUES(19, 'GRIGORE', 'GABRIELA', 0730843078, 7);

INSERT INTO ELEVI

VALUES(20, 'SANDU', 'MIRUNA', 0730643078, 8);



Inserari in tabela „BIBLIOTECA”:

INSERT INTO BIBLIOTECA

VALUES(1, 100, 45, 2);

INSERT INTO BIBLIOTECA

VALUES(2, 220, 90, 5);

INSERT INTO BIBLIOTECA

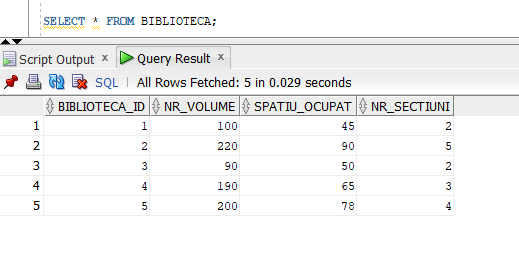
VALUES(3, 90, 50, 2);

INSERT INTO BIBLIOTECA

VALUES(4, 190, 65, 3);

INSERT INTO BIBLIOTECA

VALUES(5, 200, 78, 4);



Inserari in tabela „TEREN\_SPORT”:

INSERT INTO TEREN\_SPORT

VALUES (1, 'FOTBAL', 500);

INSERT INTO TEREN\_SPORT

VALUES (2, 'TENIS', 200);

INSERT INTO TEREN\_SPORT

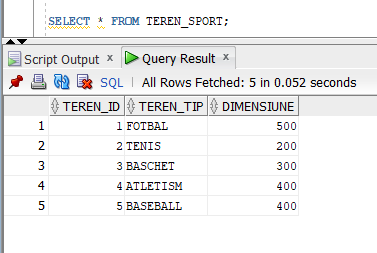
VALUES (3, 'BASCHET', 300);

INSERT INTO TEREN\_SPORT

VALUES (4, 'ATLETISM', 400);

INSERT INTO TEREN\_SPORT

VALUES (5, 'BASEBALL', 400);



Inserari in tabela „CLADIRI\_ANEXE”:

INSERT INTO CLADIRI\_ANEXE

VALUES (5, 'CAMERA DE SERVICIU', 50);

INSERT INTO CLADIRI\_ANEXE

VALUES (6, 'ATELIER', 60);

INSERT INTO CLADIRI\_ANEXE

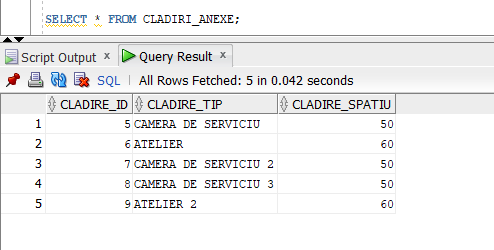
VALUES (7, 'CAMERA DE SERVICIU 2', 50);

INSERT INTO CLADIRI\_ANEXE

VALUES (8, 'CAMERA DE SERVICIU 3', 50);

INSERT INTO CLADIRI\_ANEXE

VALUES (9, 'ATELIER 2', 60);



Inserari in tabela „ARHIVA”:

INSERT INTO ARHIVA

VALUES (1, 20);

INSERT INTO ARHIVA

VALUES (2, 30);

INSERT INTO ARHIVA

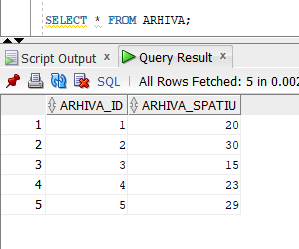
VALUES (3, 15);

INSERT INTO ARHIVA

VALUES (4, 23);

INSERT INTO ARHIVA

VALUES (5, 29);



Inserari in tabela „CABINET\_MEDICAL”:

INSERT INTO CABINET\_MEDICAL

VALUES (1, 40);

INSERT INTO CABINET\_MEDICAL

VALUES (2, 50);

INSERT INTO CABINET\_MEDICAL

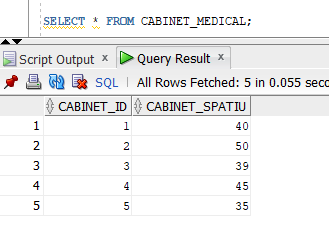
VALUES (3, 39);

INSERT INTO CABINET\_MEDICAL

VALUES (4, 45);

INSERT INTO CABINET\_MEDICAL

VALUES (5, 35);



Inserari in tabela „PROFESORI”:

INSERT INTO PROFESORI

VALUES (10, 'CIOCARLAN', 'DORINA', 0712345678, 3000, 'dorina@yahoo.com');

INSERT INTO PROFESORI

VALUES (11, 'BLOSCENCO', 'ALINA', 0714345678, 3500, 'alina34@yahoo.com');

INSERT INTO PROFESORI

VALUES (12, 'MUNTEANU', 'RARES', 0714346778, 3200, 'mrares@yahoo.com');

INSERT INTO PROFESORI

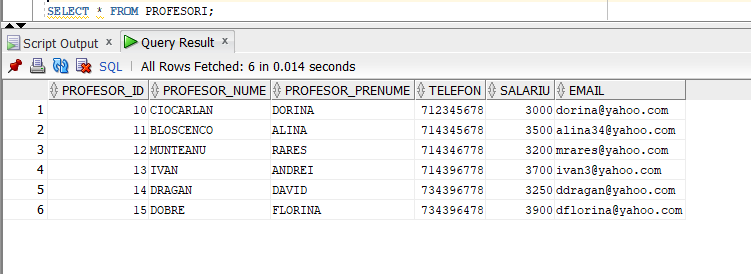
VALUES (13, 'IVAN', 'ANDREI', 0714396778, 3700, 'ivan3@yahoo.com');

INSERT INTO PROFESORI

VALUES (14, 'DRAGAN', 'DAVID', 0734396778, 3250, 'ddragan@yahoo.com');

INSERT INTO PROFESORI

VALUES (15, 'DOBRE', 'FLORINA', 0734396478, 3900, 'dflorina@yahoo.com');



Inserari in tabela „INGRIJITOR”:

INSERT INTO INGRIJITOR

VALUES (20, 'POPA', 'VASILICA', 0798765432, 2000, 'pvasilica@yahoo.com', 5);

INSERT INTO INGRIJITOR

VALUES (21, 'MOCANU', 'ALEXANDRU', 0798755432, 2500, 'MOCAALE@yahoo.com', 6);

INSERT INTO INGRIJITOR

VALUES (22, 'COJOCARU', 'PAUL', 0793255432, 2240, 'COJOPAUL@yahoo.com', 7);

INSERT INTO INGRIJITOR

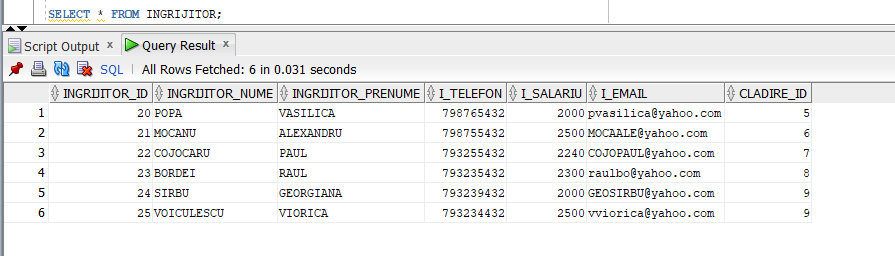
VALUES (23, 'BORDEI', 'RAUL', 0793235432, 2300, 'raulbo@yahoo.com', 8);

INSERT INTO INGRIJITOR

VALUES (24, 'SIRBU', 'GEORGIANA', 0793239432, 2000, 'GEOSIRBU@yahoo.com', 9);

INSERT INTO INGRIJITOR

VALUES (25, 'VOICULESCU', 'VIORICA', 0793234432, 2500, 'vviorica@yahoo.com', 9);



Inserari in tabela „ARHIVAR”:

INSERT INTO ARHIVAR

VALUES (30, 'DRAGNEA', 'OLGUTA', 0725737491, 4000, 'dolga@yahoo.com', 1);

INSERT INTO ARHIVAR

VALUES (31, 'GOLEA', 'VIORICA', 0775737491, 4100, 'viog@yahoo.com', 2);

INSERT INTO ARHIVAR

VALUES (32, 'MARIN', 'RUXANDRA', 0775737651, 4200, 'mruxandra@yahoo.com', 3);

INSERT INTO ARHIVAR

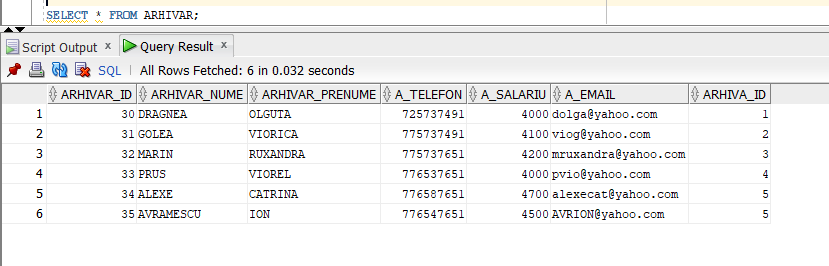
VALUES (33, 'PRUS', 'VIOREL', 0776537651, 4000, 'pvio@yahoo.com', 4);

INSERT INTO ARHIVAR

VALUES (34, 'ALEXE', 'CATRINA', 0776587651, 4700, 'alexecat@yahoo.com', 5);

INSERT INTO ARHIVAR

VALUES (35, 'AVRAMESCU', 'ION', 0776547651, 4500, 'AVRION@yahoo.com', 5);



Inserari in tabela „MEDIC”:

INSERT INTO MEDIC

VALUES (40, 'OXFORD', 'ANDREI', 0712345678, 5000, 'oxforda@yahoo.com', 1);

INSERT INTO MEDIC

VALUES (41, 'ORBAN', 'DAN', 0712344678, 5100, 'DANORBAN@yahoo.com', 2);

INSERT INTO MEDIC

VALUES (42, 'GHEORGHITA', 'MARIANA', 0732344678, 5200, 'MARIANAGHEO@yahoo.com', 3);

INSERT INTO MEDIC

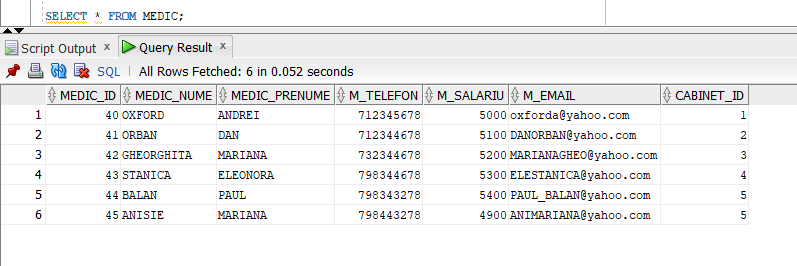
VALUES (43, 'STANICA', 'ELEONORA', 0798344678, 5300, 'ELESTANICA@yahoo.com', 4);

INSERT INTO MEDIC

VALUES (44, 'BALAN', 'PAUL', 0798343278, 5400, 'PAUL\_BALAN@yahoo.com', 5);

INSERT INTO MEDIC

VALUES (45, 'ANISIE', 'MARIANA', 0798443278, 4900, 'ANIMARIANA@yahoo.com', 5);



Inserari in tabela „BIBLIOTECAR”:

INSERT INTO BIBLIOTECAR

VALUES (1, 'POPESCU', 'ANDREI', 0712345678, 3000, 'PANDREI@YAHOO.COM', 2);

INSERT INTO BIBLIOTECAR

VALUES (2, 'POP', 'GHEORGHE', 0713345678, 3200, 'PGHEO@YAHOO.COM', 1);

INSERT INTO BIBLIOTECAR

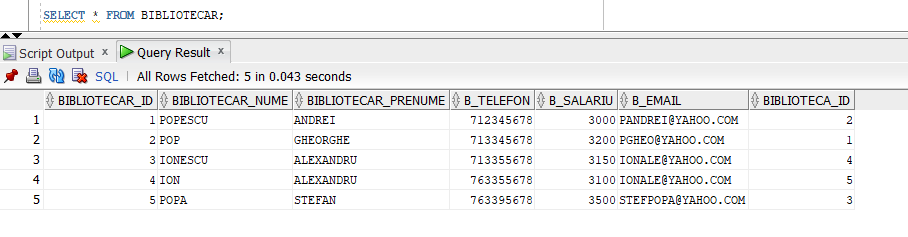
VALUES (3, 'IONESCU', 'ALEXANDRU', 0713355678, 3150, 'IONALE@YAHOO.COM', 4);

INSERT INTO BIBLIOTECAR

VALUES (4, 'ION', 'ALEXANDRU', 0763355678, 3100, 'IONALE@YAHOO.COM', 5);

INSERT INTO BIBLIOTECAR

VALUES (5, 'POPA', 'STEFAN', 0763395678, 3500, 'STEFPOPA@YAHOO.COM', 3);



Inserari in tabela asociativa „PREDA”:

INSERT INTO PREDA

VALUES (10, 1, 'MATEMATICA');

INSERT INTO PREDA

VALUES (10, 2, 'MATEMATICA');

INSERT INTO PREDA

VALUES (11, 1, 'BIOLOGIE');

INSERT INTO PREDA

VALUES (12, 3, 'CHIMIE');

INSERT INTO PREDA

VALUES (12, 4, 'CHIMIE');

INSERT INTO PREDA

VALUES (13, 5, 'FIZICA');

INSERT INTO PREDA

VALUES (12, 5, 'FIZICA');

INSERT INTO PREDA

VALUES (14, 6, 'ISTORIE');

INSERT INTO PREDA

VALUES (14, 7, 'ISTORIE');

INSERT INTO PREDA

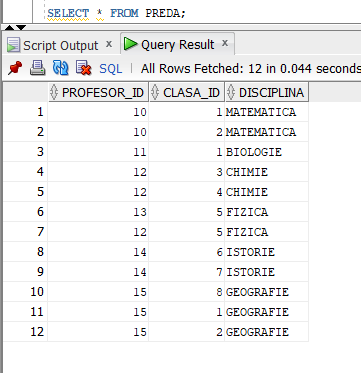
VALUES (15, 8, 'GEOGRAFIE');

INSERT INTO PREDA

VALUES (15, 1, 'GEOGRAFIE');

INSERT INTO PREDA

VALUES (15, 2, 'GEOGRAFIE');



Inserari in tabela asociativa „FOLOSESTE”:

INSERT INTO FOLOSESTE

VALUES (11, 1, 'FOTBAL');

INSERT INTO FOLOSESTE

VALUES (11, 2, 'TENIS');

INSERT INTO FOLOSESTE

VALUES (12, 1, 'FOTBAL');

INSERT INTO FOLOSESTE

VALUES (12, 3, 'BASCHET');

INSERT INTO FOLOSESTE

VALUES (11, 4, 'ATLETISM');

INSERT INTO FOLOSESTE

VALUES (11, 5, 'BASEBALL');

INSERT INTO FOLOSESTE

VALUES (12, 2, 'TENIS');

INSERT INTO FOLOSESTE

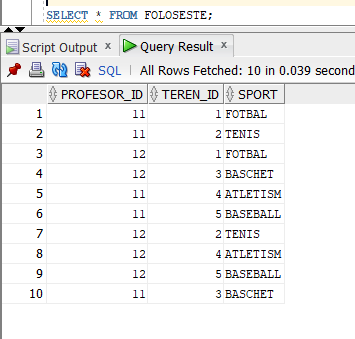
VALUES (12, 4, 'ATLETISM');

INSERT INTO FOLOSESTE

VALUES (12, 5, 'BASEBALL');

INSERT INTO FOLOSESTE

VALUES (11, 3, 'BASCHET');



1. Formulati in limbaj natural o problema pe care sa o rezolvati folosind un **subprogram stocat** care sa utilizeze doua tipuri de colectii studiate. Apelati subprogramul.

Folosind o procedura stocata ce primeste ca parametru un numar, afisati numele profesorilor si elevilor care au lungimea numelui mai mica decat numarul dat.

create or replace procedure date\_profesor(numar numeric) as

type tablou\_indexat is table of profesori.profesor\_nume%type index by binary\_integer;

p\_nume tablou\_indexat;

type tablou\_imbricat is table of elevi.elev\_nume%type;

e\_nume tablou\_imbricat;

begin

select profesor\_nume

bulk collect into p\_nume

from profesori

where length(profesor\_nume) < numar;

if p\_nume.count = 0 then

dbms\_output.put\_line('Nu a fost gasit niciun profesor');

else

for i in p\_nume.first..p\_nume.last loop

dbms\_output.put\_line(p\_nume(i));

end loop;

end if;

select elev\_nume

bulk collect into e\_nume

from elevi

where length(elev\_nume) < numar;

if e\_nume.count = 0 then

dbms\_output.put\_line('Nu a fost gasit niciun elev');

else

for i in e\_nume.first..e\_nume.last loop

dbms\_output.put\_line(e\_nume(i));

end loop;

end if;

end date\_profesor;

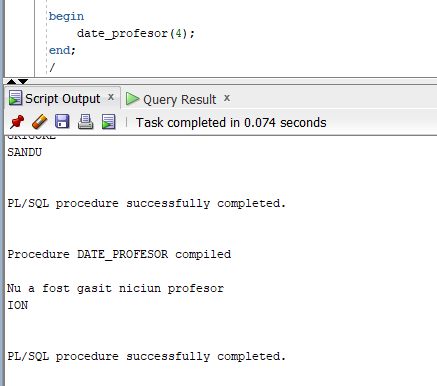
/

begin

date\_profesor(4);

end;

/



1. Formulati in limbaj natural o problema pe care sa o rezolvati folosind un **subprogram stocat** care sa utilizeze un tip de cursor studiat. Apelati subprogramul.

Folosind o procedura stocata ce primeste un numar ca parametru, aflati daca numarul specializarilor din cadrul unui profil este mai mare decat numarul dat.

create or replace procedure numar\_specializari(numar numeric) as

cursor c is

select profil\_nume nume, count(specializare\_nume) nr

from profil p, specializare s

where p.profil\_id = s.profil\_id(+)

group by profil\_nume;

begin

for i in c loop

if i.nr > numar then

dbms\_output.put\_line('Profilul ' || i.nume || ' are mai mult de ' || numar || ' specializari');

else

dbms\_output.put\_line('Profilul ' || i.nume || ' nu are mai mult de ' || numar || ' specializari');

end if;

end loop;

end numar\_specializari;

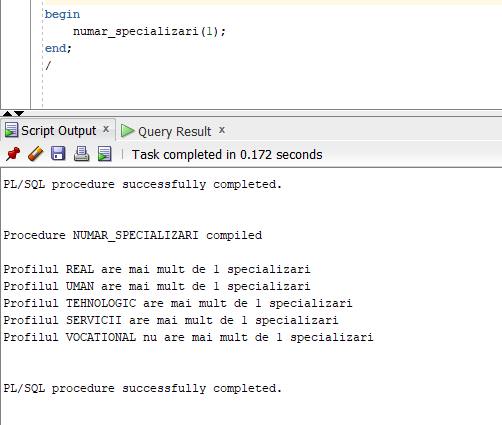
/

begin

numar\_specializari(1);

end;

/



1. Formulati in limbaj natural o problema pe care sa o rezolvati folosind un **subprogram stocat de tip functie** care sa utilizeze intr-o singura comanda SQL 3 dintre tabelele definite. Tratati toate exceptiile care pot aparea. Apelati subprogramul astfel incat sa evidentiati toate cazurile tratate.

Folosind un subprogram stocat de tip functie, afisati numele unui elev care studiaza la o specializare data ca parametru.

create or replace function nume\_elev (v\_nume specializare.specializare\_nume%type)

return varchar2 is nume\_complet elevi.elev\_nume%type;

begin

select e.elev\_nume

into nume\_complet

from elevi e join clasa c on (e.clasa\_id = c.clasa\_id)

join specializare s on (c.specializare\_id = s.specializare\_id)

where s.specializare\_nume = v\_nume;

return nume\_complet;

exception

when no\_data\_found then

raise\_application\_error(-20000, 'Nu exista elevi care studiaza la specializarea data');

when too\_many\_rows then

raise\_application\_error(-20001, 'Exista mai multi elevi');

when others then

raise\_application\_error(-20002, 'Alta eroare');

end nume\_elev;

/

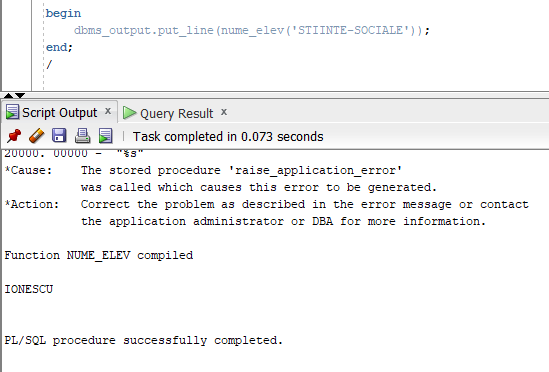
begin

dbms\_output.put\_line(nume\_elev('STIINTE-SOCIALE'));

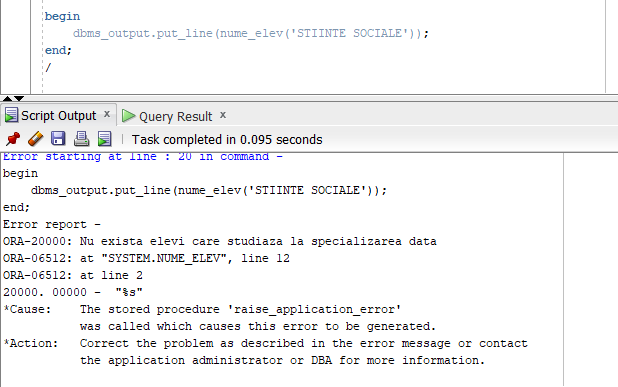
end;

/

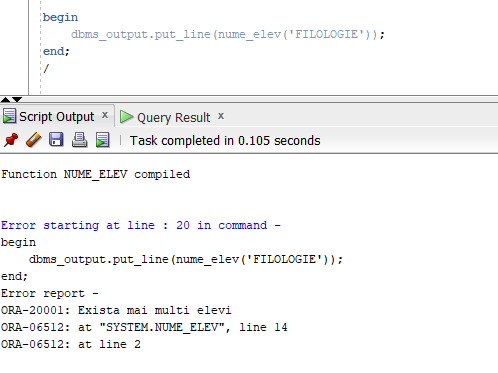
Cazul 1: programul ruleaza fara erori



Cazul 2: nu este gasita specializarea cautata



Cazul 3: intalnim mai multi elevi care studiaza la aceeasi specializare



1. Formulati in limbaj natural o problema pe care sa o rezolvati folosind un **subprogram stocat de tip procedura** care sa utilizeze intr-o singura comanda SQL 5 dintre tabelele definite. Tratati toate exceptiile care pot aparea, incluzand exceptiile NO\_DATA\_FOUND si TOO\_MANY\_ROWS. Apelati subprogramul astfel incat sa evidentiati cazurile tratate.

Folosind un subprogram stocat de tip procedura afisati un nume de profesor care preda la clasa de profil al carui nume va fi transmis ca parametru.

create or replace procedure nume\_profesor(v\_nume profil.profil\_nume%type)

is

nume profesori.profesor\_nume%type;

begin

select distinct p.profesor\_nume

into nume

from profesori p join preda pr on (p.profesor\_id = pr.profesor\_id)

join clasa c on (c.clasa\_id = pr.clasa\_id)

join specializare s on (s.specializare\_id = c.specializare\_id)

join profil pro on (pro.profil\_id = s.profil\_id)

where pro.profil\_nume = v\_nume;

dbms\_output.put\_line(nume);

exception

when no\_data\_found then

raise\_application\_error(-20000, 'Nu exista un profil cu numele dat');

when too\_many\_rows then

raise\_application\_error(-20001, 'Exista mai multi profesori care predau la acest profil');

when others then

raise\_application\_error(-20002, 'Alta eroare');

end nume\_profesor;

/

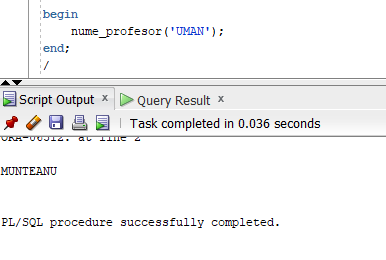
begin

nume\_profesor('UMAN');

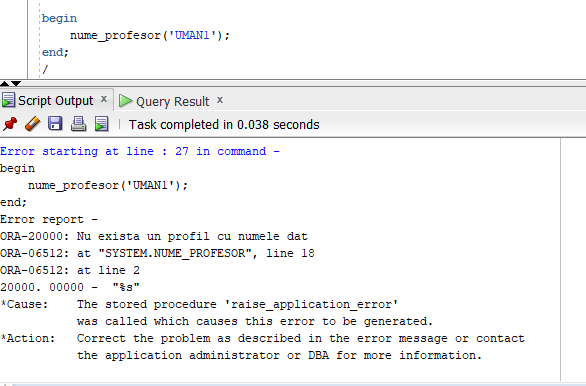
end;

/

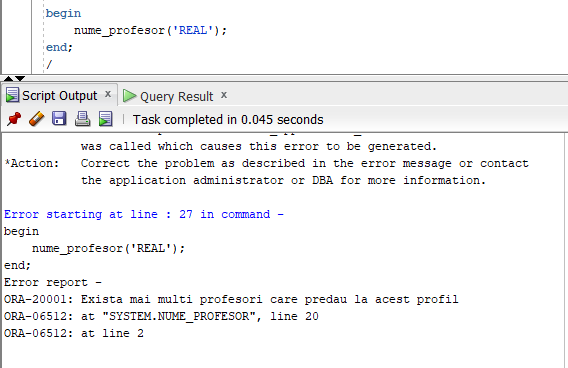
Cazul 1: programul ruleaza fara erori



Cazul 2: nu este gasit profilul pentru care se face cautarea



Cazul 3: exista mai multi profesori care predau la profilul transmis ca parametru



1. Definiti un *trigger* de tip LMD la nivel de comanda. Declansati *trigger*-ul.

Definiti un declansator prin care sa nu se permita inserarea unei noi arhive daca aceasta are spatiul mai mic de 10 metri patrati.

create or replace trigger trig10

before insert or update or delete on arhiva

begin

if arhiva\_spatiu < 10 then

raise\_application\_error(-20001, 'Tabelul nu poate fi actualizat');

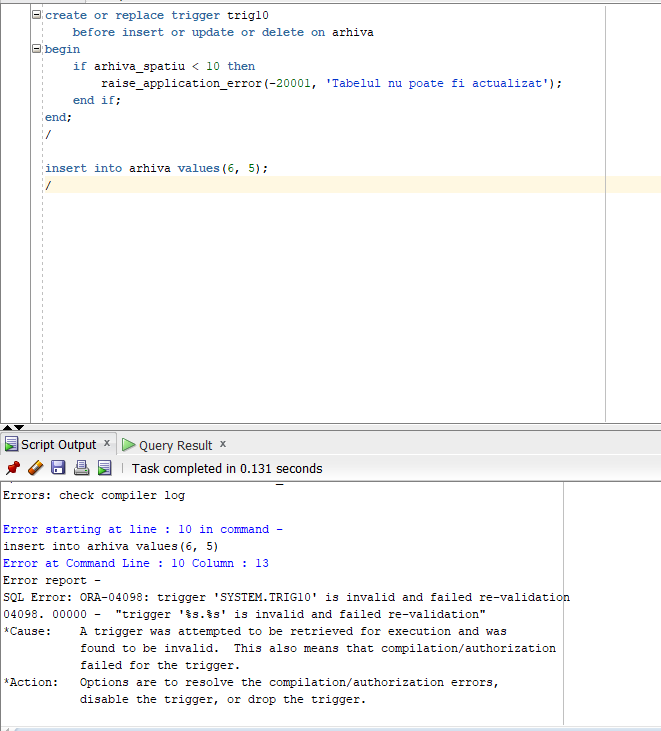
end if;

end;

/

insert into arhiva values(6, 5);

/



1. Definiti un *trigger* de tip LMD la nivel de linie. Declansati *trigger*-ul.

Definiti un declansator la nivel de linie prin care sa nu se permita micsorarea salariilor angajatilor din tabelul „ARHIVAR”.

create or replace trigger trig11

before update of a\_salariu on arhivar

for each row

begin

if (:new.a\_salariu < :old.a\_salariu) then

raise\_application\_error(-20002, 'Salariul nu poate fi micsorat');

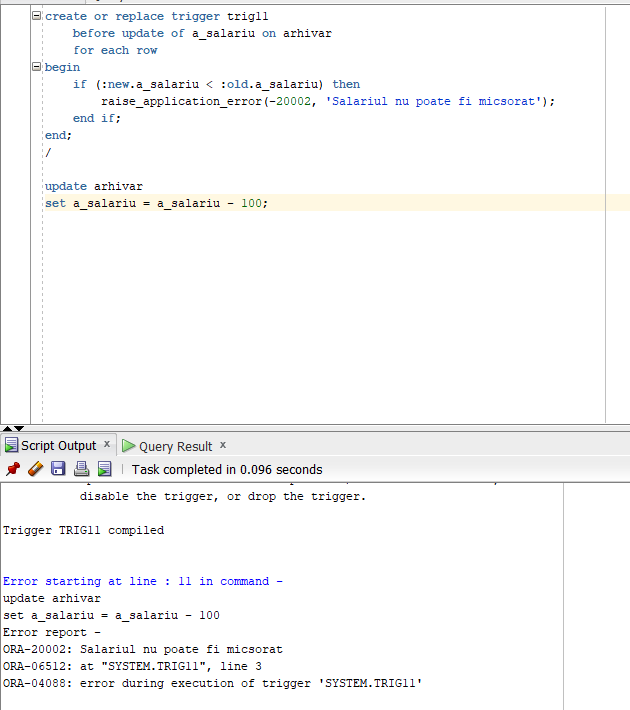
end if;

end;

/

update arhivar

set a\_salariu = a\_salariu - 100;



1. Definiti un *trigger* de tip LDD. Declansati *trigger*-ul.

Definiti un declansator sistem la nivel de baza de date care sa introduca date referitoare la erorile aparute.

create table erori (user\_id nvarchar2(100), nume\_baza nvarchar2(100), erori nvarchar2(2000), data date);

create or replace trigger trig12

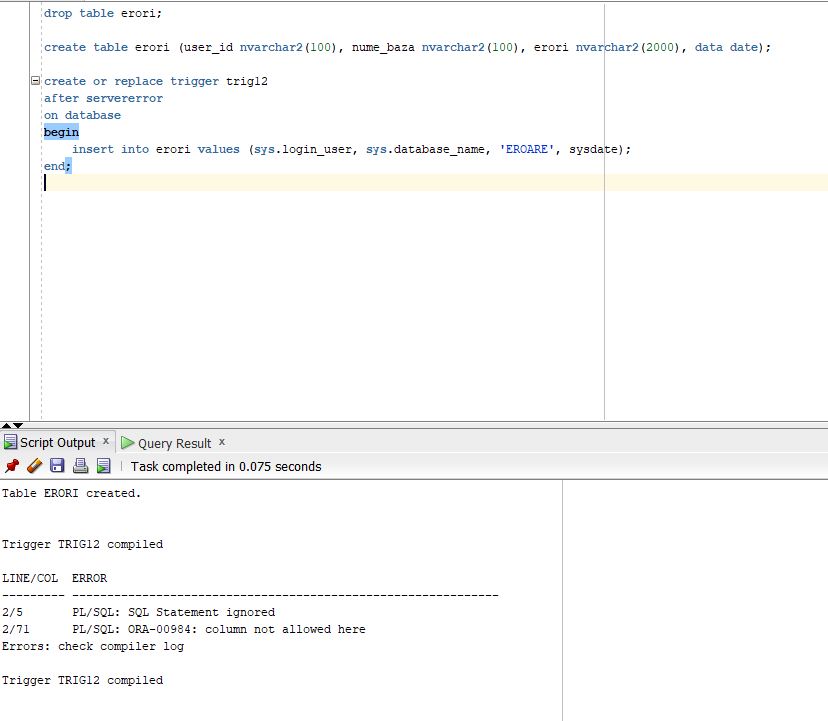
after servererror

on database

begin

insert into erori values (sys.login\_user, sys.database\_name, 'EROARE', sysdate);

end;



1. Definiti un pachet care sa contina toate obiectele definite in cadrul proiectului.

create or replace package pachet13 as

-- 6

procedure date\_profesor(numar numeric);

-- 7

procedure numar\_specializari(numar numeric);

-- 8

function nume\_elev (v\_nume specializare.specializare\_nume%type)

return varchar2;

-- 9

procedure nume\_profesor(v\_nume profil.profil\_nume%type);

end;

/

create or replace package body pachet13 as

-- 6

procedure date\_profesor(numar numeric) as

type tablou\_indexat is table of profesori.profesor\_nume%type index by binary\_integer;

p\_nume tablou\_indexat;

type tablou\_imbricat is table of elevi.elev\_nume%type;

e\_nume tablou\_imbricat;

begin

select profesor\_nume

bulk collect into p\_nume

from profesori

where length(profesor\_nume) < numar;

if p\_nume.count = 0 then

dbms\_output.put\_line('Nu a fost gasit niciun profesor');

else

for i in p\_nume.first..p\_nume.last loop

dbms\_output.put\_line(p\_nume(i));

end loop;

end if;

select elev\_nume

bulk collect into e\_nume

from elevi

where length(elev\_nume) < numar;

if e\_nume.count = 0 then

dbms\_output.put\_line('Nu a fost gasit niciun elev');

else

for i in e\_nume.first..e\_nume.last loop

dbms\_output.put\_line(e\_nume(i));

end loop;

end if;

end date\_profesor;

-- 7

procedure numar\_specializari(numar numeric) as

cursor c is

select profil\_nume nume, count(specializare\_nume) nr

from profil p, specializare s

where p.profil\_id = s.profil\_id(+)

group by profil\_nume;

begin

for i in c loop

if i.nr > numar then

dbms\_output.put\_line('Profilul ' || i.nume || ' are mai mult de ' || numar || ' specializari');

else

dbms\_output.put\_line('Profilul ' || i.nume || ' nu are mai mult de ' || numar || ' specializari');

end if;

end loop;

end numar\_specializari;

-- 8

function nume\_elev (v\_nume specializare.specializare\_nume%type)

return varchar2 is nume\_complet elevi.elev\_nume%type;

begin

select e.elev\_nume

into nume\_complet

from elevi e join clasa c on (e.clasa\_id = c.clasa\_id)

join specializare s on (c.specializare\_id = s.specializare\_id)

where s.specializare\_nume = v\_nume;

return nume\_complet;

exception

when no\_data\_found then

raise\_application\_error(-20000, 'Nu exista elevi care studiaza la specializarea data');

when too\_many\_rows then

raise\_application\_error(-20001, 'Exista mai multi elevi');

when others then

raise\_application\_error(-20002, 'Alta eroare');

end nume\_elev;

-- 9

procedure nume\_profesor(v\_nume profil.profil\_nume%type)

is

nume profesori.profesor\_nume%type;

begin

select distinct p.profesor\_nume

into nume

from profesori p join preda pr on (p.profesor\_id = pr.profesor\_id)

join clasa c on (c.clasa\_id = pr.clasa\_id)

join specializare s on (s.specializare\_id = c.specializare\_id)

join profil pro on (pro.profil\_id = s.profil\_id)

where pro.profil\_nume = v\_nume;

dbms\_output.put\_line(nume);

exception

when no\_data\_found then

raise\_application\_error(-20000, 'Nu exista un profil cu numele dat');

when too\_many\_rows then

raise\_application\_error(-20001, 'Exista mai multi profesori care predau la acest profil');

when others then

raise\_application\_error(-20002, 'Alta eroare');

end nume\_profesor;

end pachet13;

/

