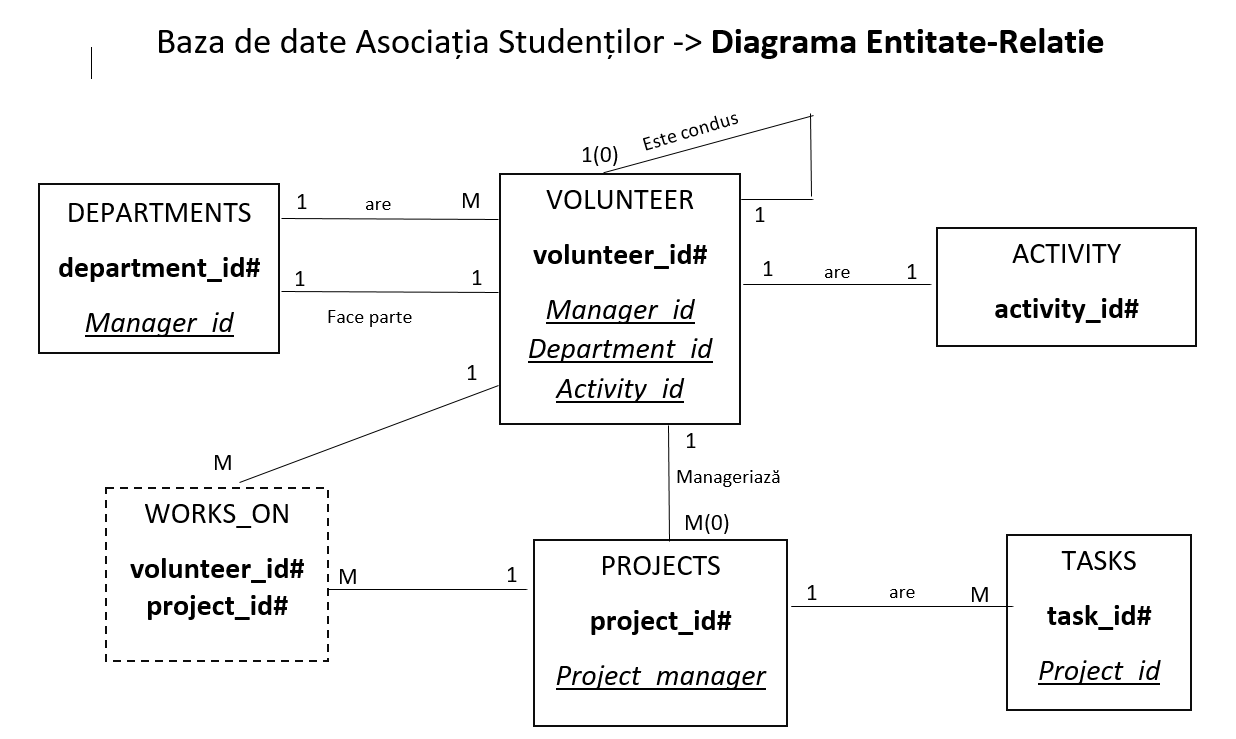
Baza de date Asocația Studenților

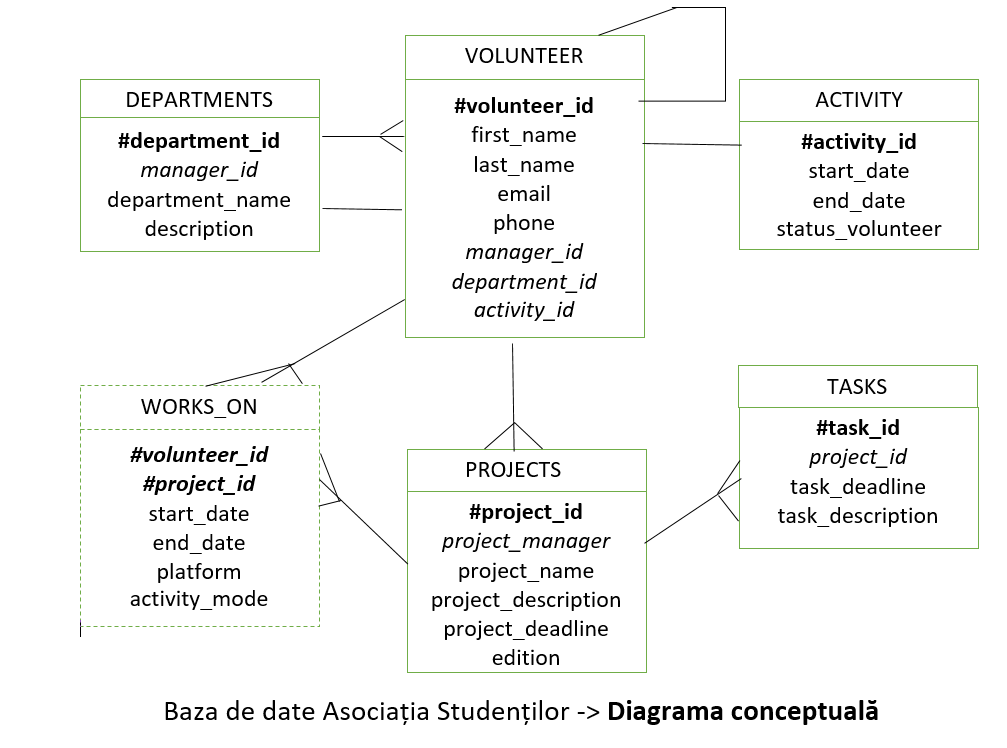
Dima Oana-Teodora

Grupa 241

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

După 11 ani de existență a Asociației Studenților la Matematică și Informatică, președintele hotărăște că trebuie să existe o îmbunătățire considerabilă în managerierea proiectelor, a voluntarilor și a task-urilor. Își dorește ca toate să se afle într-un singur loc, evidența lor să fie ținută în timp real, voluntarii să-și poată îndeplini task-urile mai ușor, iar directorii de departament să scape de rapoartele de activitate întocmite în fiecare lună. De asemenea urmărește păstrarea istoricului asociației, a proiectelor desfășurate de-a lungul timpului, împreună cu edițiile evenimentelor, dar și a membrilor onorifici. Astfel că, în decursul acestui an, apelează la Departamentul IT al FMI pentru crearea unei baze de date corespunzătoare. În urma unei ședințe de brainstorming, membrii departamentului ajung la concluzia că vor fi necesare 6 tabele dintre care unul va fi de legatură. Datele importante ale voluntarilor care se doresc a fi păstrate sunt: numele, prenumele, numărul de telefon și email-ul . Fiecare voluntar poate fi : activ, inactiv, alumn sau președinte și face parte dintr-un singur departament . Fiecare departament este condus de un singur manager, iar fiecare manager la rândul sau este condus de președinte. Există o listă de task-uri comune care se aplică proiectelor. De asemenea, există și proiecte care sunt la prima ediție, dar majoritatea proiectelor au avut ediții și în anii precedenți, păstrând formula de desfășurare. Mai mulți voluntari pot lucra la mai multe proiecte, mai ales că unele dintre acestea se pot suprapune, iar fiecare proiect este manageriat de un voluntar, dar acelasi voluntar poate manageria mai multe proiecte.

2. Realizați diagrama entitate-relație (ERD).

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

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

5. Adăugați informații coerente în tabelele create (minim 3-5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru tabela asociativă).

create table volunteer

( volunteer\_id number(4)constraint PK\_VOLUNTEER\_ID primary key,

first\_name varchar(20) not null,

last\_name varchar(20) not null,

email varchar(20) not null,

phone varchar(10),

manager\_id number(4),

department\_id number(4),

activity\_id number(4)

);

create table departments

( department\_id number(4)constraint PK\_DEPARTMENT\_ID primary key,

manager\_id number(4),

department\_name varchar(20),

description varchar(50)

);

--adaug cheile externe

--legatura cu departments

alter table volunteer

add constraint FK\_DEPARTEMNT\_ID

foreign key (department\_id) references departments(department\_id);

--self-joinul de la volunteer

alter table volunteer

add constraint FK\_VOLUNTEER\_ID

foreign key(manager\_id) references volunteer(volunteer\_id);

--departments cu volunteer prin manager\_id

alter table departments

add constraint FK\_MANAGER\_ID

foreign key(manager\_id) references volunteer(volunteer\_id);

create table activity

(

activity\_id number(4)constraint PK\_ACTIVITY\_ID primary key,

start\_date Date,

end\_date Date,

status\_volunteer varchar(10) default 'activ'

);

--adaug legatura cu volunteer

alter table volunteer

add constraint FK\_ACTIVITY\_ID

foreign key (activity\_id) references activity(activity\_id);

create table tasks

(

task\_id number(4)constraint PK\_TASK\_ID primary key,

project\_id number(4),

task\_deadline Date,

task\_description varchar(50)

);

create table projects

(

project\_id number(4)constraint PK\_PROJECT\_ID primary key,

project\_manager number(4),

project\_name varchar(20),

project\_description varchar(50),

project\_deadline date

);

alter table projects

add(edition number(2) not null);

--legatura projects tasks

alter table tasks

add constraint FK\_PROJECT\_ID

foreign key (project\_id) references projects(project\_id)

on delete cascade;

--legatura volunteer- projects

alter table projects

add constraint FK\_PROJECT\_MANAGER

foreign key (project\_manager) references volunteer(volunteer\_id)

on delete cascade;

create table works\_on

(

volunteer\_id number(4),

project\_id number(4),

start\_date date,

end\_date date,

platform varchar(20),

activity\_mode varchar(20) not null

);

--cheia primara compusa

alter table works\_on

add constraint PK\_WORKS\_ON

primary key(volunteer\_id, project\_id);

--legatura cu tabela volunteer

alter table works\_on

add constraint FK\_WROKS\_ON\_VOLUNTEER

foreign key (volunteer\_id) references volunteer(volunteer\_id)

on delete cascade;

--legatura cu projects

alter table works\_on

add constraint FK\_WORKS\_ON\_PROJECTS

foreign key (project\_id) references projects(project\_id)

on delete cascade;

--5 departamente

/\*

10 HR

11 D&PR

12 FR

13 Proiecte

14 Edu

\*/

--activity\_id numerele de la 101 la 107

insert into activity

values (101, '1-OCT-2019', null, 'activ' );

insert into activity

values (102, '1-OCT-2019', null, 'inactiv' );

insert into activity

values (103, '1-OCT-2009', '1-OCT-2019', 'alumn' );

insert into activity

values (104, '1-OCT-2019', null, 'presedinte' );

insert into activity

values (105, '1-OCT-2018', '1-NOV-2020', 'alumn' );

insert into activity

values (106, '1-NOV-2020', null, 'activ' );

insert into activity

values (107, '1-NOV-2019', null, 'inactiv' );

--departments

insert into departments(department\_id)

values(10);

insert into departments(department\_id)

values(11);

insert into departments(department\_id)

values(12);

insert into departments(department\_id)

values(13);

insert into departments(department\_id)

values(14);

insert into volunteer(volunteer\_id, last\_name, first\_name, email, phone)

values (1, 'Silistru', 'Delia-Stefania', 'silistru@s.unibuc.ro', '0123456789');

insert into volunteer(volunteer\_id, last\_name, first\_name, email, phone)

values (3, 'Dorneanu', 'Alina-Mihaela', 'dorneanu@s.unibuc.ro', '0113156189');

insert into volunteer(volunteer\_id, last\_name, first\_name, email, phone)

values (4, 'Popa', 'Iulia-Andreea', 'popa@s.unibuc.ro', '0122456722');

insert into volunteer(volunteer\_id, last\_name, first\_name, email, phone)

values (5, 'Bigan', 'Marian-Antonio', 'bigan@s.unibuc.ro', '0100456789');--manager Proiecte

insert into volunteer(volunteer\_id, last\_name, first\_name, email, phone)

values (6, 'Nimara', 'Dan-Gabriel', 'nimara@s.unibuc.ro', '0183486788');

insert into volunteer(volunteer\_id, last\_name, first\_name, email, phone)

values (7, 'Dirstariu', 'Daria', 'dirstar@s.unibuc.ro', '0125556789');

insert into volunteer(volunteer\_id, last\_name, first\_name, email, phone)

values (8, 'Daineanu', 'Denis', 'daineanu@s.unibuc.ro', '0123456000' );--manager design

insert into volunteer(volunteer\_id, last\_name, first\_name, email, phone)

values (9, 'Marcu', 'Florian', 'marcu@s.unibuc.ro', '0124444489' );

insert into volunteer(volunteer\_id, last\_name, first\_name, email, phone)

values (10, 'Fusneica', 'Florin', 'fusneica@s.unibuc.ro', '0120456789' );

insert into volunteer(volunteer\_id, last\_name, first\_name, email, phone)

values (11, 'Badescu', 'Gabriel', 'badescu@s.unibuc.ro', '0123456785' );--manager edu

insert into volunteer(volunteer\_id, last\_name, first\_name, email, phone)

values (12, 'Pavalasc', 'Irina', 'pavalasc@s.unibuc.ro', '0122226789');

insert into volunteer(volunteer\_id, last\_name, first\_name, email, phone)

values (13, 'Bertalan', 'Victor', 'bertalan@s.unibuc.ro', '0123456711' );

insert into volunteer(volunteer\_id, last\_name, first\_name, email, phone)

values (14, 'Negulescu', 'Radu', 'negules@s.unibuc.ro', '0123400089');--manager fr

insert into volunteer(volunteer\_id, last\_name, first\_name, email, phone)

values (15, 'Guta', 'Razvan', 'guta@s.unibuc.ro', '0123666789');

insert into volunteer(volunteer\_id, last\_name, first\_name, email, phone)

values (16, 'Vijulan', 'Stefania-Claudia', 'vijulan@s.unibuc.ro', '0123333789' );

insert into volunteer(volunteer\_id, last\_name, first\_name, email, phone)

values (2, 'Neagu', 'Andrei-Stefan', 'neagu@s.unibuc.ro', '0123223789' );

update departments

set manager\_id=1 , department\_name ='Human Resources', description='recrutarea noilor membrii si de team building'

where department\_id=10;

update departments

set manager\_id=8 , department\_name ='Design si PR', description='imaginea asociatiei'

where department\_id=11;

update departments

set manager\_id=14 , department\_name ='Fundraising', description='se ocupa de gasirea sponsorilor'

where department\_id=12;

update departments

set manager\_id=5 , department\_name ='Proiecte', description='se ocupa de orgnizarea petrecerilor'

where department\_id=13;

update departments

set manager\_id=11 , department\_name ='Educational', description='se ocupa de relatiile cu profesorii'

where department\_id=14;

update volunteer

set activity\_id=104

where volunteer\_id=2;--presedinte

update volunteer

set manager\_id=2, department\_id=10, activity\_id=101

where volunteer\_id=1;--manager HR

update volunteer

set manager\_id=1, department\_id=10, activity\_id=101

where volunteer\_id=3;

update volunteer

set manager\_id=1, department\_id=10, activity\_id=101

where volunteer\_id=4;

update volunteer

set manager\_id=2, department\_id=13, activity\_id=101

where volunteer\_id=5;

update volunteer

set manager\_id=5, department\_id=13, activity\_id=106

where volunteer\_id=6;

update volunteer

set manager\_id=5, department\_id=13, activity\_id=107

where volunteer\_id=7;

update volunteer

set manager\_id=2, department\_id=11, activity\_id=106

where volunteer\_id=8;

update volunteer

set manager\_id=8, department\_id=11, activity\_id=102

where volunteer\_id=9;

update volunteer

set manager\_id=8, department\_id=11, activity\_id=102

where volunteer\_id=10;

update volunteer

set manager\_id=2, department\_id=14, activity\_id=106

where volunteer\_id=11;

update volunteer

set manager\_id=11, department\_id=14, activity\_id=107

where volunteer\_id=12;

update volunteer

set manager\_id=11, department\_id=14, activity\_id=101

where volunteer\_id=13;

update volunteer

set manager\_id=2, department\_id=12, activity\_id=101

where volunteer\_id=14;

update volunteer

set manager\_id=14, department\_id=12, activity\_id=106

where volunteer\_id=15;

update volunteer

set manager\_id=14, department\_id=12, activity\_id=106

where volunteer\_id=16;

--inserare in works\_on, projects, tasks

--proiecte:

--Recrutari 21

--Arta'n dar 22

--TeamBuilding 23

--Cariere 24

--SmartHack 25

--UBSU 26

insert into projects

values(21, 3, 'Recrutari', 'alegerea noilor membrii', '1-DEC-2020', 11);

insert into projects

values(22, 9, 'Arta-n dar', 'sustinere cauza carotabila', '25-DEC-2020', 5);

insert into projects

values(23,10 , 'TeamBuilding', 'creeare comunitate', '1-OCT-2021', 2);

insert into projects

values(24,16 , 'Cariere', 'oferire internship', '10-MAY-2021', 5);

insert into projects

values(25, 13, 'SmartHack', 'Hackathon FMI', '8-NOV-2021', 20);

insert into projects

values(26,7 , 'UBSU', 'atragerea viitorilor studenti', '2-MAY-2020', 3);

insert into projects

values(27,3 , 'UBSU', 'atragerea viitorilor studenti', '2-MAY-2019', 2);

insert into projects

values(28,4 , 'UBSU', 'atragerea viitorilor studenti', '2-MAY-2018', 1);

insert into tasks

values( 200, 21, '29-NOV-2020', 'creare formular de inscriere');

insert into tasks

values( 201, 22, '29-NOV-2020', 'impodobire sediu');

insert into tasks

values( 202, 22, '1-DEC-2020', 'printare afise');

insert into tasks

values( 203, 22, '29-NOV-2020', 'materiale stand');

insert into tasks

values( 207, 24, '10-JAN-2021', 'organizare prezentari');

insert into tasks

values( 208, 25, '3-NOV-2021', 'printat diplome');

insert into tasks

values( 209, 25, '3-NOV-2021', 'inchiriat sala');

insert into tasks

values( 210, 25, '1-NOV-2021', 'contact firma catering');

insert into tasks

values( 211, 25, '28-OCT-2021', 'trimis invitatii');

insert into tasks

values( 212, 25, '29-OCT-2021', 'contactare echipe');

insert into tasks

values( 213, 26, '29-JAN-2020', 'contact participanti');

insert into tasks

values( 214, 26, '1-MAY-2020', 'imprimare tricouri');

insert into tasks

values( 215, 27, '26-JAN-2019', 'contact ASUB');

insert into tasks

values( 216, 28, '25-FEB-2018', 'contact profesori');

insert into tasks

values( 217, 28, '1-MAY-2018', 'organizare camere');

select \* from tasks;

select \* from projects;

insert into works\_on

values( 4, 21, '1-NOV-2020', '30-NOV-2020', 'teams', 'online');

insert into works\_on

values( 3, 21, '2-NOV-2020', '30-NOV-2020', 'Discord', 'online');

insert into works\_on

values( 5, 21, '3-NOV-2020', '29-NOV-2020', 'teams', 'online');

insert into works\_on

values( 6, 21, '1-NOV-2020', '28-NOV-2020', 'teams', 'online');

insert into works\_on

values( 1, 22, '10-NOV-2020', '24-DEC-2020', null, 'fizic');

insert into works\_on

values( 13, 22, '10-NOV-2020', '30-NOV-2020', null, 'fizic');

insert into works\_on

values( 15, 23, '11-SEP-2021', '31-OCT-2021', null, 'fizic');

insert into works\_on

values( 16, 23, '12-SEP-2021', '29-OCT-2021', 'Discord', 'online');

insert into works\_on

values( 10, 25, '1-OCT-2021', '8-NOV-2021', null, 'fizic');

insert into works\_on

values( 11, 25, '5-OCT-2021', '5-NOV-2021', 'teams', 'online');

insert into works\_on

values( 11, 23, '11-SEP-2021', '31-OCT-2021', null, 'fizic');

insert into works\_on

values( 4, 22, '15-NOV-2020', '20-DEC-2020', 'Webex', 'online');

insert into projects

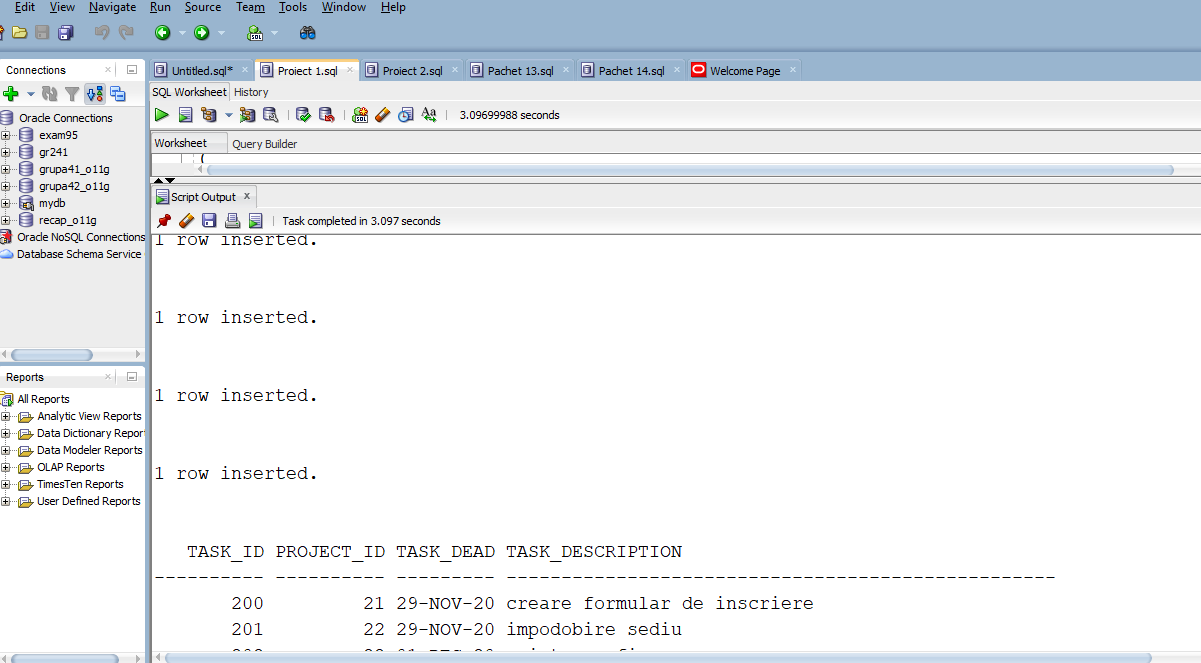
values(29, null, 'ProjectASMI', 'introducere boboci in organizare proiecte', null, 1);

insert into departments

values(15, null,'IT' , null);

insert into projects values

(30, null, 'Caravana Edu', 'promovare facultate', null, 1);

commit;

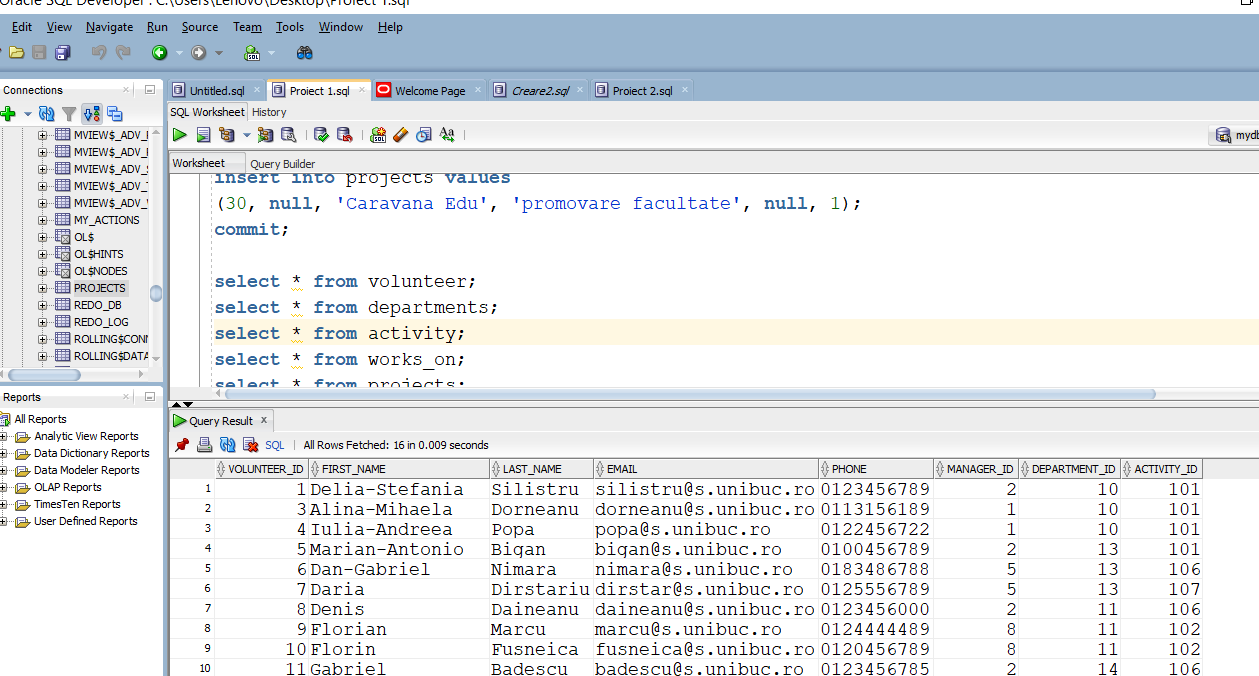
select \* from volunteer;

select \* from departments;

select \* from activity;

select \* from works\_on;

select \* from projects;

select \* from tasks;

6. Definiți un subprogram stocat care să utilizeze un tip de colecție studiat. Apelați subprogramul.

Vreau să șterg și să afișez (împreună cu numele proiecteleor de care aparțin) task-urile a căror deadline a expirat, apoi vreau să adaug aceleași task-uri cu deadline-ul modificat (aceeași zi, lună, dar anul următor).

* Colecții utilizate: **tablou indexat**, **înregistrare**.
* Subprogram de tip **procedură** stocat.

CREATE OR REPLACE PROCEDURE punctul\_6

IS

Type rec IS RECORD

( cod tasks.task\_id%type,

prj tasks.project\_id%type,

deadline tasks.task\_deadline%type,

descriere tasks.task\_description%type);

TYPE tabel IS TABLE OF rec INDEX BY PLS\_INTEGER;

t tabel :=tabel();

nume projects.project\_name%type;

BEGIN

--stergerea cu retinerea informatiilor

DELETE FROM tasks

WHERE task\_deadline< sysdate

RETURNING task\_id, project\_id, task\_deadline, task\_description BULK COLLECT INTO t ;

DBMS\_OUTPUT.PUT\_LINE('Lista de task-uri expirate: ');

FOR i IN t.first..t.last LOOP

--iau numele proiectului de care apartine task-ul

SELECT project\_name

INTO nume

FROM projects

WHERE project\_id=t(i).prj;

DBMS\_OUTPUT.PUT\_LINE('Taskul: '|| t(i).descriere ||' apartine de proiectul: '||nume );

END LOOP;

FOR i IN t.first..t.last LOOP

--reactualizarea tabelei de task-uri

INSERT INTO tasks

VALUES(t(i).cod, t(i).prj, t(i).deadline+365\*(2021-to\_number(to\_char(t(i).deadline, 'yyyy'))), t(i).descriere);

END LOOP;

EXCEPTION

--in cazul in care nu avem task-uri in care

--deadlineul a expirat

WHEN value\_error THEN

DBMS\_OUTPUT.PUT\_LINE('Nu sunt date de actualizat');

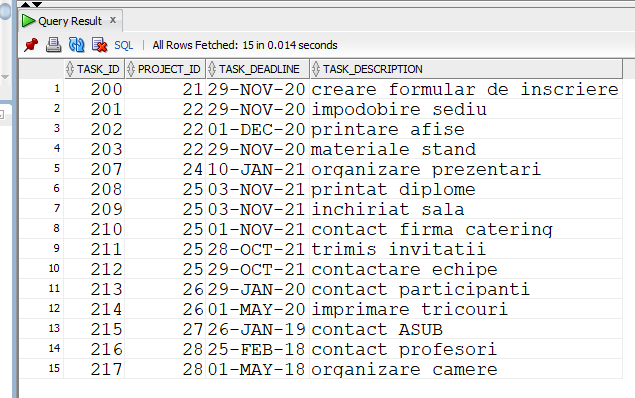
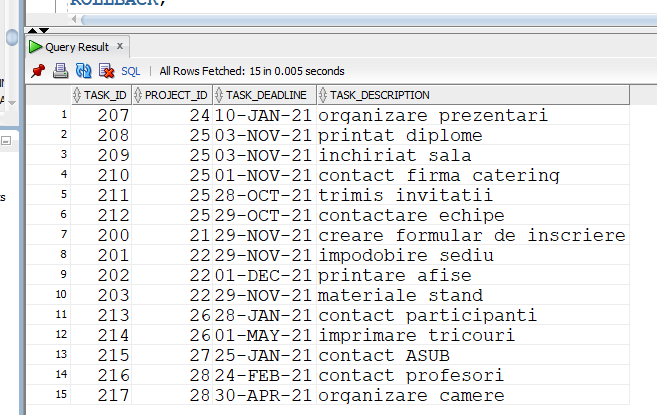
END punctul\_6;

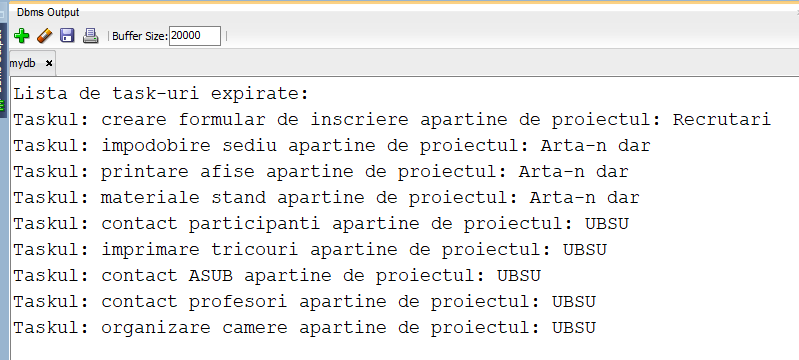
/

EXECUTE punctul\_6;

SELECT \* FROM tasks;

ROLLBACK;

**BEFORE:** **AFTER:**



7. Definiți un subprogram stocat care să utilizeze un tip de cursor studiat. Apelați subprogramul.

În funcție de o opțiune introdusă de la tastatură (una dintre șirurile activ, inactiv, alumn), deschid un cursor astfel încât să se regăsească:

- lista voluntarilor activi împreună cu numele departamentelor din care fac parte(pentru opțiunea activ);

- lista voluntarilor inactivi împreună cu data în care au intrat în asociație(pentru opțiunea inactiv);

- lista alumnilor împreună cu anul în care au ieșit din asociație(pentru opțiunea alumn).

* **Cursor dinamic**
* Subprogram de tip **procedură** stocat

CREATE OR REPLACE PROCEDURE punctul\_7(optiune VARCHAR2)

IS

TYPE rec1 IS RECORD

( nume volunteer.last\_name%type,

prenume volunteer.first\_name%type,

nume\_dep departments.department\_name%type);

TYPE rec23 IS RECORD

( nume volunteer.last\_name%type,

prenume volunteer.first\_name%type,

activ activity.start\_date%type);

TYPE tip1 IS REF CURSOR RETURN rec1;

TYPE tip23 IS REF CURSOR RETURN rec23;

cer1 tip1;

cer23 tip23;

v1 rec1;

v23 rec23;

ok\_found BOOLEAN:=false;

BEGIN

IF optiune ='activ' THEN

OPEN cer1 FOR

SELECT last\_name, first\_name, department\_name

FROM volunteer v JOIN departments d ON (v.department\_id=d.department\_id)

JOIN activity a On (v.activity\_id=a.activity\_id)

WHERE LOWER(status\_volunteer)='activ';

--afisare

LOOP

FETCH cer1 INTO v1;

EXIT WHEN cer1%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(v1.nume ||' ' || v1.prenume|| ' ' || v1.nume\_dep);

ok\_found:=true;

END LOOP;

CLOSE cer1;

ELSIF optiune = 'inactiv' THEN

OPEN cer23 FOR

SELECT last\_name, first\_name, start\_date

FROM volunteer v JOIN activity a ON (v.activity\_id=a.activity\_id)

WHERE LOWER(status\_volunteer)='inactiv';

--afisare

LOOP

FETCH cer23 into v23;

EXIT WHEN cer23%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(v23.nume||' ' || v23.prenume||' ' || v23.activ);

ok\_found:=true;

END LOOP;

CLOSE cer23;

ELSIF optiune = 'alumn' THEN

OPEN cer23 FOR

SELECT last\_name, first\_name, end\_date

FROM volunteer v JOIN activity a ON (v.activity\_id=a.activity\_id)

WHERE LOWER(status\_volunteer)='alumn';

--afisare

LOOP

FETCH cer23 into v23;

EXIT WHEN cer23%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(v23.nume||' ' || v23.prenume ||' '|| v23.activ);

ok\_found:=true;

END LOOP;

CLOSE cer23;

ELSE

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

END IF;

IF ok\_found=false THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista date');

END IF;

END punctul\_7;

/

--Apel:

DECLARE

--activ , inactiv sau alumn

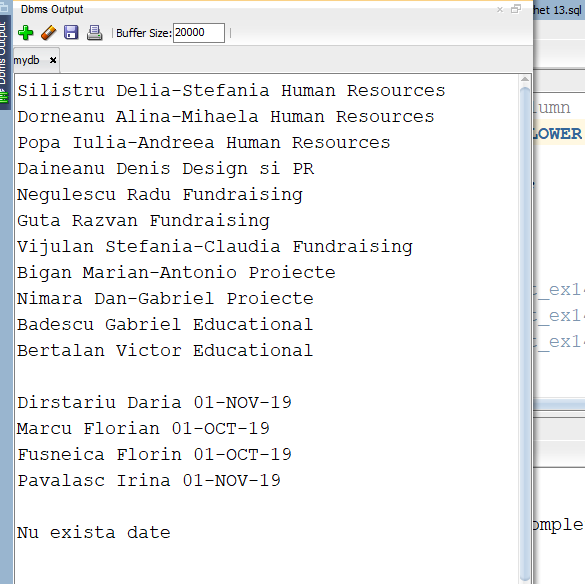
optiune VARCHAR2(8) := LOWER('&p\_optiune');

BEGIN

punctul\_7(optiune);

END;

/



8. Definiți un subprogram stocat de tip funcție care să utilizeze 3 dintre tabelele definite. Tratați toate excepțiile care pot apărea. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

Vreau să afișez pentru un departament dat, raportul dintre numărul total de membrii activi și numărul total de membrii din departament și pentru departamentele care au doar membrii activi, să afișez lista membrilor.

dep de Proiecte nu are doar membrii activi

dep 10 Human Resources, 12 Fundraising - au doar membrii activi

pt dep IT am impartire la 0 (0 membrii in departament)=>zero\_divide

+lista vida de membrii => collection\_is\_null

* Tabele: **departments, volunteer, activity**
* Subprogram de tip **funcție** stocat
* Excepții tratate: **no\_data\_found, zero\_divide, collection\_is\_null**

CREATE OR REPLACE FUNCTION punctul\_8 (dep departments.department\_name%type)

RETURN VARCHAR2

Is

nr\_vol NUMBER(3);

nr\_activ NUMBER(3);

TYPE vect IS VARRAY(100) OF volunteer.first\_name%type;

v vect;

nr NUMBER(3);

id\_dep departments.department\_id%type;

raport FLOAT:=0;

BEGIN

SELECT COUNT(\*)

INTO nr\_vol

FROM volunteer v JOIN activity a On (v.activity\_id=a.activity\_id)

WHERE LOWER(status\_volunteer)='activ';

SELECT d.department\_id,COUNT(volunteer\_id)

INTO id\_dep, nr

FROM departments d LEFT JOIN volunteer v ON (d.department\_id=v.department\_id)

WHERE LOWER(department\_name)=lower(dep)

GROUP BY d.department\_id;

SELECT COUNT(volunteer\_id)

INTO nr\_activ

FROM volunteer v JOIN activity a ON (v.activity\_id=a.activity\_id)

WHERE department\_id=id\_dep AND LOWER(status\_volunteer)='activ';

raport := round(nr\_vol/nr,2);

IF nr=nr\_activ THEN

v:= vect();

SELECT first\_name BULK COLLECT INTO v

FROM volunteer v JOIN activity a ON (v.activity\_id=a.activity\_id)

WHERE department\_id=id\_dep AND LOWER(status\_volunteer)='activ';

END IF;

FOR i IN v.first..v.last LOOP

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

END LOOP;

RETURN ('Raportul este: ' ||to\_char(raport));

EXCEPTION

WHEN no\_data\_found THEN

RETURN 'Ai gresit numele departamentului! Mai incearca :(';

WHEN zero\_divide THEN

RETURN 'Hopa! Departamentul nu are voluntari (impartire la 0)';

WHEN collection\_is\_null THEN

RETURN 'Departamentul nu are numai voluntari activi';

END punctul\_8;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Pentru collection\_is\_null');

DBMS\_OUTPUT.PUT\_LINE(punctul\_8('Proiecte')); --collection\_is\_null

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Pentru zero\_divide');

DBMS\_OUTPUT.PUT\_LINE(punctul\_8('IT')); --zero\_divide

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Nu arunca eroare');

DBMS\_OUTPUT.PUT\_LINE(punctul\_8('Fundraising')); --ok

END;

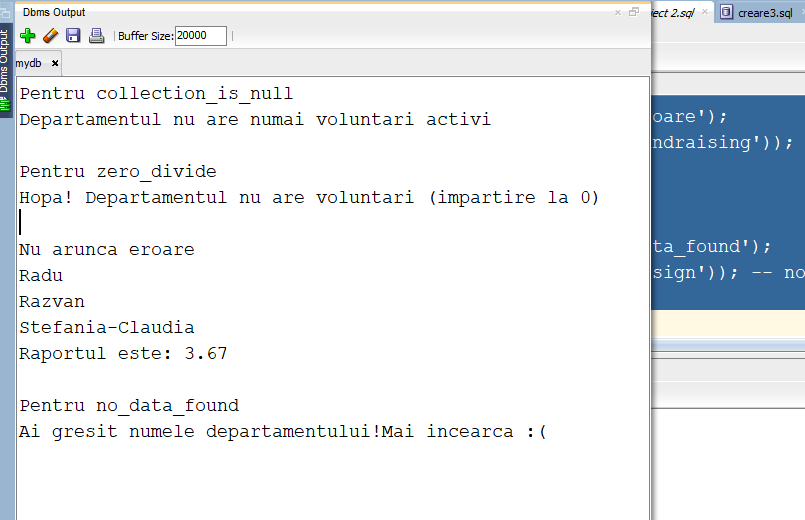
/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Pentru no\_data\_found');

DBMS\_OUTPUT.PUT\_LINE(punctul\_8('Design')); -- no\_data\_found

END;/



9. Definiți un subprogram stocat de tip procedură care să utilizeze 5 dintre tabelele definite. Tratați toate excepțiile care pot apărea. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

Pentru un nume de voluntar introdus de la tastatură afișez lista de task-uri, numele și ediția proiectului pe care îl conduce, departamentul din care face parte și dacă este sau nu alumn (se iau în considerare doar voluntarii care conduc primul lor proiect).

Teambuilding 2 nu are task-uri alocate si este condus de Fusneica=> value\_error

voluntarul Dorneanu a project manageriat 2 proiecte=> too\_many\_rows

voluntarul Badescu nu a project manageriat => no data found

* Tabele: **projects, tasks, volunteer, departments, activity**
* Subprogram tip **procedură** stocat

CREATE OR REPLACE PROCEDURE punctul\_9 (voluntar volunteer.last\_name%type)

IS

proiect projects.project\_name%type;

editie projects.edition%type;

TYPE tabel IS TABLE OF tasks%rowtype INDEX BY PLS\_INTEGER;

taskuri tabel;

departament departments.department\_name%type;

status activity.status\_volunteer%type;

cod projects.project\_id%type;

BEGIN

SELECT department\_name, status\_volunteer

INTO departament, status

FROM departments d JOIN volunteer v ON (d.department\_id=v.department\_id)

JOIN activity a ON (v.activity\_id=a.activity\_id)

WHERE UPPER(last\_name)=UPPER(voluntar);

SELECT project\_name, edition, project\_id

INTO proiect, editie, cod

FROM projects

WHERE project\_manager =(SELECT volunteer\_id

FROM volunteer

WHERE UPPER(last\_name)=UPPER(voluntar));

DBMS\_OUTPUT.PUT('Voluntarul: '||voluntar|| ' face parte din departamentul: ' ||departament);

IF LOWER(status)='alumn' THEN

DBMS\_OUTPUT.PUT\_LINE(' si este alumn');

ELSE

DBMS\_OUTPUT.PUT\_LINE(' si nu este alumn');

End If;

SELECT \* BULK COLLECT INTO taskuri

FROM tasks

WHERE project\_id = cod;

DBMS\_OUTPUT.PUT\_LINE('Proiectul: ' || proiect ||' editia: ' || editie );

FOR i IN taskuri.first..taskuri.last LOOP

DBMS\_OUTPUT.PUT\_LINE(taskuri(i).task\_description);

END LOOP;

EXCEPTION

WHEN value\_error THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista task-uri alocate pentru acest proiect');

WHEN no\_data\_found THEN

DBMS\_OUTPUT.PUT\_LINE('Nu conduce niciun proiect');

WHEN too\_many\_rows THEN

DBMS\_OUTPUT.PUT\_LINE('Ati dat un voluntar care nu se afla la primul proiect manageriat');

END punctul\_9;

/

BEGIN

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

punctul\_9('Bertalan');--fara eroare

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Pentru value\_error');

punctul\_9('Fusneica');-- value\_error

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Pentru too\_many\_rows');

punctul\_9('Dorneanu');--too\_many\_rows

END;

/

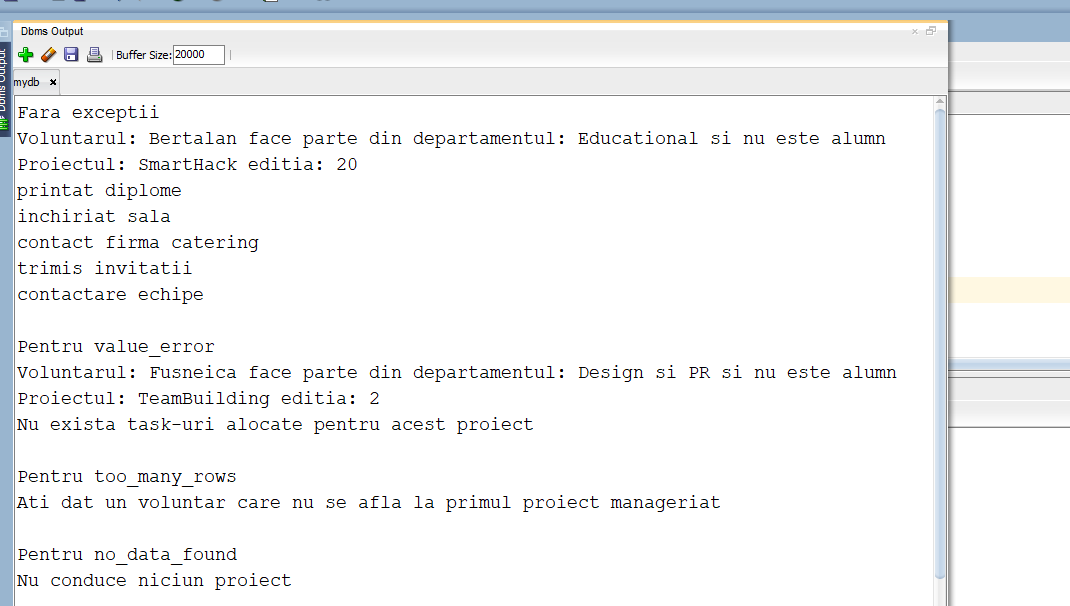
BEGIN

DBMS\_OUTPUT.PUT\_LINE('Pentru no\_data\_found');

punctul\_9('Badescu');--no\_data\_found

END;

/



10. Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.

Vreau să setez un trigger care nu lasă să fie inserat un nou proiect în tabelul de proiecte, dacă există cel puțin un poroiect care încă nu are setat un deadline.

CREATE OR REPLACE TRIGGER trigger\_ex10

BEFORE INSERT ON projects

DECLARE

nr NUMBER:=0;

BEGIN

SELECT COUNT(project\_id)

INTO nr

FROM projects

WHERE project\_deadline IS NULL;

IF (nr>0) AND INSERTING THEN

RAISE\_APPLICATION\_ERROR(-20001,'Nu poti adauga un nou proiect pana

nu ai stabilit deadline-ul celor existente!');

END IF;

END;

/

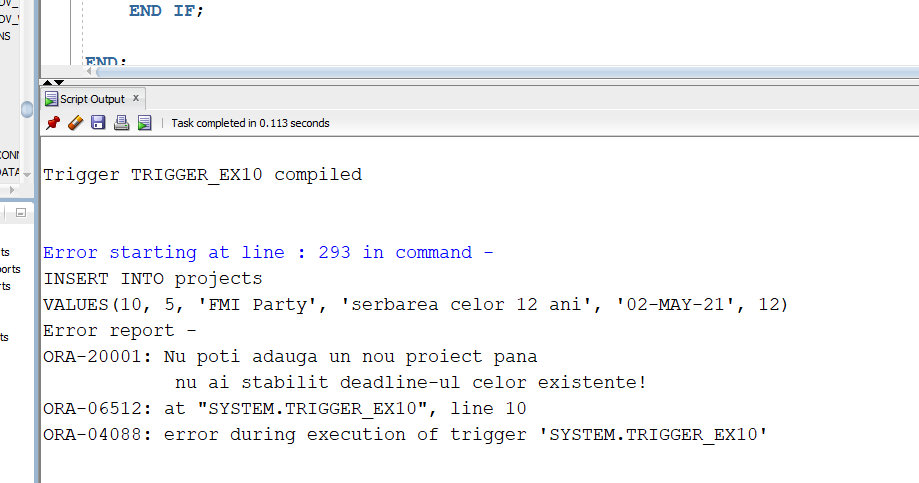
--declansare trigger

INSERT INTO projects

VALUES(10, 5, 'FMI Party', 'serbarea celor 12 ani', '02-MAY-21', 12);

--stergere trigger

DROP TRIGGER trigger\_ex10;



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

Vreau să setez un trigger care șterge automat voluntarii inactivit în momentul în care data de final a activității (end\_date) a unui voluntar cu status inactiv, este updatată din null în data curentă(înseamnă că voluntarul este dat afară din asociație). Afișez și informații despre aceștia.

* Mă folosesc de **on delete cascade** de la cheile externe.
* Fac o **procedură** prin care șterg voluntarii.

CREATE OR REPLACE PROCEDURE sterge\_voluntar(activ\_id activity.activity\_id%type)

IS

TYPE rec IS RECORD

(nume volunteer.last\_name%type,

prenume volunteer.first\_name%type,

telefon volunteer.phone%type);

TYPE tabel IS TABLE OF rec;

t tabel:=tabel();

BEGIN

DELETE FROM volunteer

WHERE activity\_id=activ\_id

RETURNING last\_name, first\_name, phone BULK COLLECT INTO t;

FOR i IN t.first..t.last Loop

DBMS\_OUTPUT.PUT\_LINE(t(i).nume||' '|| t(i).prenume||' '|| t(i).telefon);

END LOOP;

END;

/

CREATE OR REPLACE TRIGGER trigger\_ex11

BEFORE UPDATE of end\_date ON activity

FOR EACH ROW

WHEN (NEW.end\_date=sysdate)

BEGIN

sterge\_voluntar(:old.activity\_id);

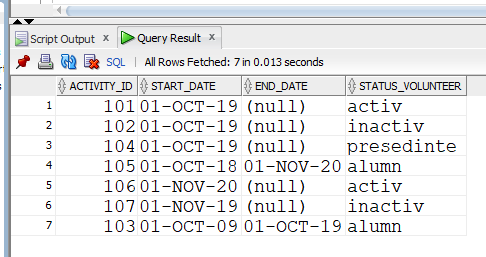
END;

/

UPDATE activity

SET end\_date=sysdate

WHERE LOWER(status\_volunteer)='inactiv';

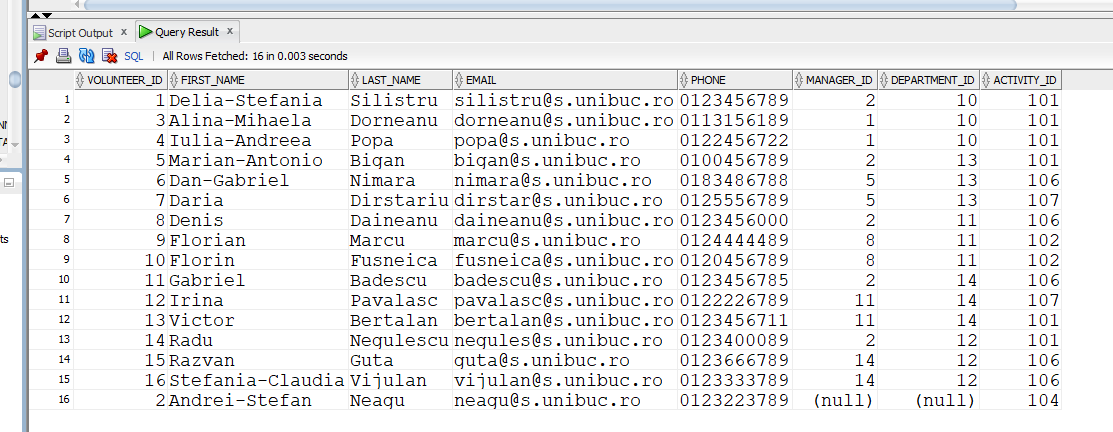
SELECT \* FROM activity;

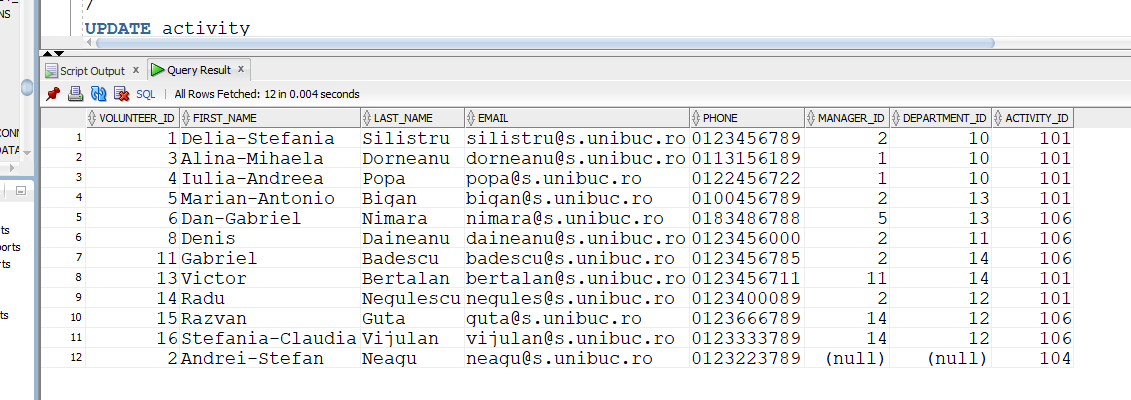
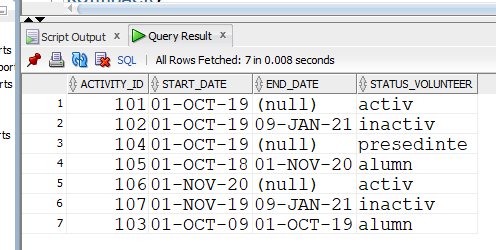
SELECT \* FROM volunteer;

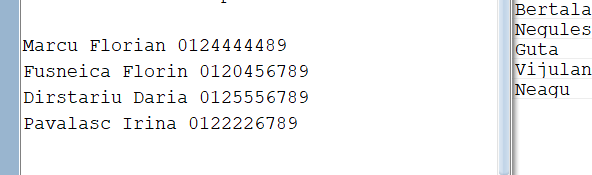
ROLLBACK;

DROP TRIGGER trigger\_ex11;

**BEFORE:**



**AFTER**



12. Definiți un trigger de tip LDD. Declanșați trigger-ul.

Vreau să setez un trigger care, în tabelul my\_actions pune doar comenzile care au avut loc fără erori, celelalte fiind doar avertizate printr-un mesaj.

CREATE TABLE MY\_ACTIONS

(utilizator VARCHAR2(30),

eveniment VARCHAR2(20),

data TIMESTAMP(3));

SELECT \* FROM my\_actions;

CREATE OR REPLACE TRIGGER trigger\_ex12

AFTER CREATE OR DROP OR ALTER or SERVERERROR ON DATABASE

BEGIN

If(DBMS\_UTILITY.FORMAT\_ERROR\_STACK is null)THEN

INSERT INTO my\_actions

VALUES (SYS.LOGIN\_USER, SYS.SYSEVENT,SYSTIMESTAMP);

ELSE

RAISE\_APPLICATION\_ERROR(-20000,'Au aparut erori in aplicatie :(');

END IF;

END;

/

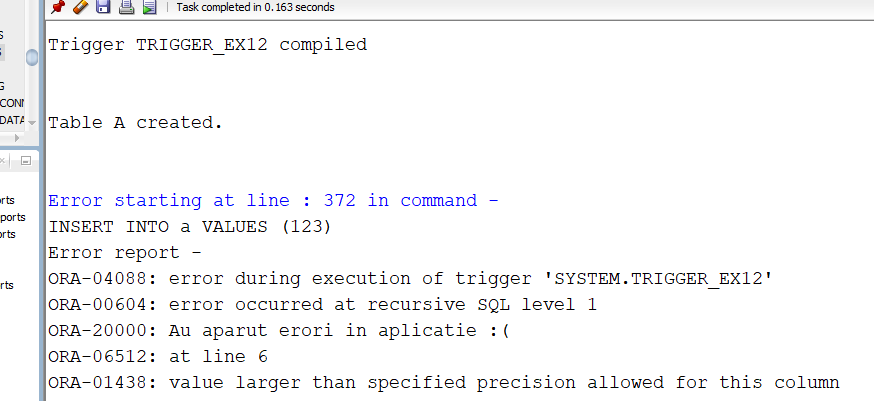
DROP TRIGGER trigger\_ex12;

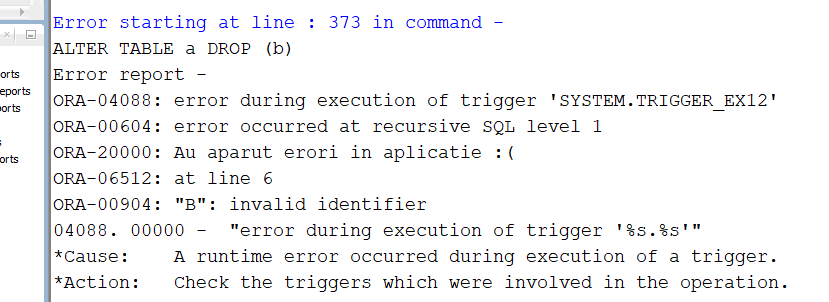
/

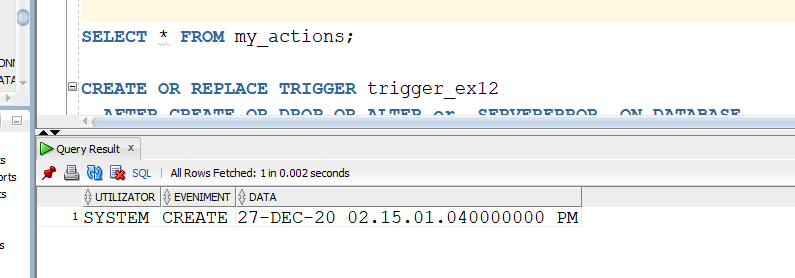
CREATE TABLE a (id NUMBER(2));

INSERT INTO a VALUES (123);

ALTER TABLE a DROP (b);







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

CREATE OR REPLACE PACKAGE pachet\_ex13 AS

PROCEDURE punctul\_6;

PROCEDURE punctul\_7(optiune VARCHAR2);

FUNCTION punctul\_8 (dep departments.department\_name%type) RETURN VARCHAR2;

PROCEDURE punctul\_9 (voluntar volunteer.last\_name%type);

PROCEDURE sterge\_voluntar(activ\_id activity.activity\_id%type);

END pachet\_ex13;

/

CREATE OR REPLACE PACKAGE BODY pachet\_ex13 AS

PROCEDURE punctul\_6

IS

TYPE rec IS RECORD

( cod tasks.task\_id%type,

prj tasks.project\_id%type,

deadline tasks.task\_deadline%type,

descriere tasks.task\_description%type);

TYPE tabel IS TABLE OF rec INDEX BY PLS\_INTEGER;

t tabel :=tabel();

nume projects.project\_name%type;

BEGIN

--stergerea cu retinerea informatiilor

DELETE FROM tasks

WHERE task\_deadline< sysdate

RETURNING task\_id, project\_id, task\_deadline, task\_description BULK COLLECT INTO t ;

DBMS\_OUTPUT.PUT\_LINE('Lista de task-uri expirate: ');

FOR i IN t.first..t.last LOOP

--iau numele proiectului de care apartine task-ul

SELECT project\_name

INTO nume

FROM projects

WHERE project\_id=t(i).prj;

DBMS\_OUTPUT.PUT\_LINE('Taskul: '|| t(i).descriere ||' apartine de proiectul: '||nume );

END LOOP;

FOR i IN t.first..t.last LOOP

--reactualizarea tabelei de task-uri

INSERT INTO tasks

VALUES(t(i).cod, t(i).prj, t(i).deadline+365\*(2021-to\_number(to\_char(t(i).deadline, 'yyyy'))), t(i).descriere);

END LOOP;

EXCEPTION

--in cazul in care nu avem task-uri in care

--deadlineul a expirat

WHEN value\_error THEN

DBMS\_OUTPUT.PUT\_LINE('Nu sunt date de actualizat');

END punctul\_6;

PROCEDURE punctul\_7(optiune VARCHAR2)

IS

TYPE rec1 IS RECORD

( nume volunteer.last\_name%type,

prenume volunteer.first\_name%type,

nume\_dep departments.department\_name%type);

TYPE rec23 IS RECORD

( nume volunteer.last\_name%type,

prenume volunteer.first\_name%type,

activ activity.start\_date%type);

TYPE tip1 IS REF CURSOR RETURN rec1;

TYPE tip23 IS REF CURSOR RETURN rec23;

cer1 tip1;

cer23 tip23;

v1 rec1;

v23 rec23;

ok\_found BOOLEAN:=false;

BEGIN

IF optiune ='activ' THEN

OPEN cer1 FOR

SELECT last\_name, first\_name, department\_name

FROM volunteer v JOIN departments d ON (v.department\_id=d.department\_id)

JOIN activity a On (v.activity\_id=a.activity\_id)

WHERE LOWER(status\_volunteer)='activ';

--afisare

LOOP

FETCH cer1 INTO v1;

EXIT WHEN cer1%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(v1.nume ||' ' || v1.prenume|| ' ' || v1.nume\_dep);

ok\_found:=true;

END LOOP;

CLOSE cer1;

ELSIF optiune = 'inactiv' THEN

OPEN cer23 FOR

SELECT last\_name, first\_name, start\_date

FROM volunteer v JOIN activity a ON (v.activity\_id=a.activity\_id)

WHERE LOWER(status\_volunteer)='inactiv';

--afisare

LOOP

FETCH cer23 into v23;

EXIT WHEN cer23%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(v23.nume||' ' || v23.prenume||' ' || v23.activ);

ok\_found:=true;

END LOOP;

CLOSE cer23;

ELSIF optiune = 'alumn' THEN

OPEN cer23 FOR

SELECT last\_name, first\_name, end\_date

FROM volunteer v JOIN activity a ON (v.activity\_id=a.activity\_id)

WHERE LOWER(status\_volunteer)='alumn';

--afisare

LOOP

FETCH cer23 into v23;

EXIT WHEN cer23%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(v23.nume||' ' || v23.prenume ||' '|| v23.activ);

ok\_found:=true;

END LOOP;

CLOSE cer23;

ELSE

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

END IF;

IF ok\_found=false THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista date');

END IF;

END punctul\_7;

FUNCTION punctul\_8 (dep departments.department\_name%type)

RETURN VARCHAR2

Is

nr\_vol NUMBER(3);

nr\_activ NUMBER(3);

TYPE vect IS VARRAY(100) OF volunteer.first\_name%type;

v vect;

nr NUMBER(3);

id\_dep departments.department\_id%type;

raport FLOAT:=0;

BEGIN

SELECT COUNT(\*)

INTO nr\_vol

FROM volunteer v JOIN activity a On (v.activity\_id=a.activity\_id)

WHERE LOWER(status\_volunteer)='activ';

SELECT d.department\_id,COUNT(volunteer\_id)

INTO id\_dep, nr

FROM departments d LEFT JOIN volunteer v ON (d.department\_id=v.department\_id)

WHERE LOWER(department\_name)=lower(dep)

GROUP BY d.department\_id;

SELECT COUNT(volunteer\_id)

INTO nr\_activ

FROM volunteer v JOIN activity a ON (v.activity\_id=a.activity\_id)

WHERE department\_id=id\_dep AND LOWER(status\_volunteer)='activ';

raport := round(nr\_vol/nr,2);

IF nr=nr\_activ THEN

v:= vect();

SELECT first\_name BULK COLLECT INTO v

FROM volunteer v JOIN activity a ON (v.activity\_id=a.activity\_id)

WHERE department\_id=id\_dep AND LOWER(status\_volunteer)='activ';

END IF;

FOR i IN v.first..v.last LOOP

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

END LOOP;

RETURN ('Raportul este: ' ||to\_char(raport));

EXCEPTION

WHEN no\_data\_found THEN

RETURN 'Ai gresit numele departamentului! Mai incearca :(';

WHEN zero\_divide THEN

RETURN 'Hopa! Departamentul nu are voluntari (impartire la 0)';

WHEN collection\_is\_null THEN

RETURN 'Departamentul nu are numai voluntari activi';

END punctul\_8;

PROCEDURE punctul\_9 (voluntar volunteer.last\_name%type)

Is

proiect projects.project\_name%type;

editie projects.edition%type;

TYPE tabel IS TABLE OF tasks%rowtype INDEX BY PLS\_INTEGER;

taskuri tabel;

departament departments.department\_name%type;

status activity.status\_volunteer%type;

cod projects.project\_id%type;

BEGIN

SELECT department\_name, status\_volunteer

INTO departament, status

FROM departments d JOIN volunteer v ON (d.department\_id=v.department\_id)

JOIN activity a ON (v.activity\_id=a.activity\_id)

WHERE UPPER(last\_name)=UPPER(voluntar);

SELECT project\_name, edition, project\_id

INTO proiect, editie, cod

FROM projects

WHERE project\_manager =(SELECT volunteer\_id

FROM volunteer

WHERE UPPER(last\_name)=UPPER(voluntar));

DBMS\_OUTPUT.PUT('Voluntarul: '||voluntar|| ' face parte din departamentul: ' ||departament);

IF LOWER(status)='alumn' THEN

DBMS\_OUTPUT.PUT\_LINE(' si este alumn');

ELSE

DBMS\_OUTPUT.PUT\_LINE(' si nu este alumn');

End If;

SELECT \* BULK COLLECT INTO taskuri

FROM tasks

WHERE project\_id = cod;

DBMS\_OUTPUT.PUT\_LINE('Proiectul: ' || proiect ||' editia: ' || editie );

FOR i IN taskuri.first..taskuri.last LOOP

DBMS\_OUTPUT.PUT\_LINE(taskuri(i).task\_description);

END LOOP;

EXCEPTION

WHEN value\_error THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista task-uri alocate pentru acest proiect');

WHEN no\_data\_found THEN

DBMS\_OUTPUT.PUT\_LINE('Nu conduce niciun proiect');

WHEN too\_many\_rows THEN

DBMS\_OUTPUT.PUT\_LINE('Ati dat un voluntar care nu se afla la primul proiect manageriat');

END punctul\_9;

PROCEDURE sterge\_voluntar(activ\_id activity.activity\_id%type)

IS

TYPE rec IS RECORD

(nume volunteer.last\_name%type,

prenume volunteer.first\_name%type,

telefon volunteer.phone%type);

TYPE tabel IS TABLE OF rec;

t tabel:=tabel();

BEGIN

DELETE FROM volunteer

WHERE activity\_id=activ\_id

RETURNING last\_name, first\_name, phone BULK COLLECT INTO t;

FOR i IN t.first..t.last Loop

DBMS\_OUTPUT.PUT\_LINE(t(i).nume||' '|| t(i).prenume||' '|| t(i).telefon);

END LOOP;

END sterge\_voluntar;

END pachet\_ex13;

/

--apel pt ex 6

EXECUTE pachet\_ex13.punctul\_6;

SELECT \* FROM tasks;

ROLLBACK;

--apel pt ex 7

DECLARE

--activ , inactiv sau alumn

optiune VARCHAR2(8) := LOWER('&p\_optiune');

BEGIN

pachet\_ex13.punctul\_7(optiune);

END;

/

--apel pt ex 8

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Pentru collection\_is\_null');

DBMS\_OUTPUT.PUT\_LINE(pachet\_ex13.punctul\_8('Proiecte')); --collection\_is\_null

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

DBMS\_OUTPUT.PUT\_LINE('Pentru zero\_divide');

DBMS\_OUTPUT.PUT\_LINE(pachet\_ex13.punctul\_8('IT')); --zero\_divide

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

DBMS\_OUTPUT.PUT\_LINE('Nu arunca eroare');

DBMS\_OUTPUT.PUT\_LINE(pachet\_ex13.punctul\_8('Fundraising')); --ok

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

DBMS\_OUTPUT.PUT\_LINE('Pentru no\_data\_found');

DBMS\_OUTPUT.PUT\_LINE(pachet\_ex13.punctul\_8('Design')); -- no\_data\_found

END;

/

--apel pt ex 9

BEGIN

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

pachet\_ex13.punctul\_9('Bertalan');--fara eroare

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

DBMS\_OUTPUT.PUT\_LINE('Pentru value\_error');

pachet\_ex13.punctul\_9('Fusneica');-- value\_error

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

DBMS\_OUTPUT.PUT\_LINE('Pentru too\_many\_rows');

pachet\_ex13.punctul\_9('Dorneanu');--too\_many\_rows

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

DBMS\_OUTPUT.PUT\_LINE('Pentru no\_data\_found');

pachet\_ex13.punctul\_9('Badescu');--no\_data\_found

END;

/

--triggerul de la ex 11 se foloseste de

--procedura sterge voluntari

CREATE OR REPLACE TRIGGER trigger\_ex11

BEFORE UPDATE of end\_date ON activity

FOR EACH ROW

WHEN (NEW.end\_date=sysdate)

BEGIN

pachet\_ex13.sterge\_voluntar(:old.activity\_id);

END;

/

UPDATE activity

SET end\_date=sysdate

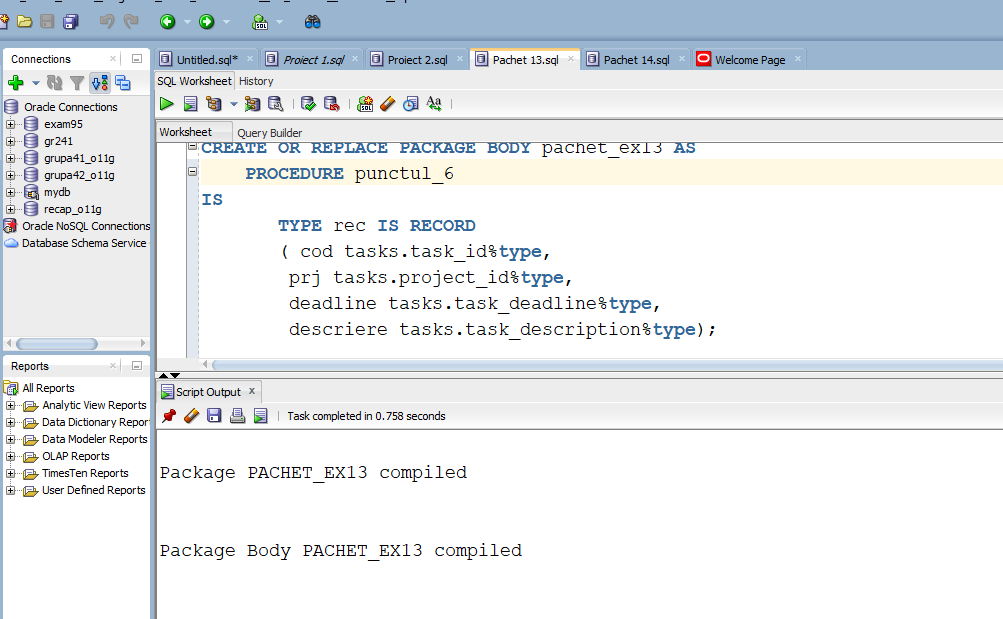
WHERE LOWER(status\_volunteer)='inactiv';

SELECT \* FROM activity;

SELECT \* FROM volunteer;

ROLLBACK;

DROP TRIGGER trigger\_ex11;



14. Definiți un pachet care să includă tipuri de date complexe și obiecte necesare pentru acțiuni integrate.

Pentru fiecare voluntar vreau să rețin și să afișez lista proiectelor la care a participat cu anumite mențiuni. Dacă este voluntar activ și are numărul proiectelor la care a participat >=2, atunci îi afșez alături de nume, titlul de membru onorific, altfel în cazul în care este voluntar inactiv și a intrat de cel putin 2 ani in asociație îl șterg din baza de date doar dacă nr proiectelor la care a participat este <= 1 și îi afisez un mesaj corespunzător.

* Tip de date complex: **vectori de vectori**

CREATE OR REPLACE PACKAGE pachet\_info\_voluntari AS

PROCEDURE creare;

FUNCTION verificare\_activ(i NUMBER) RETURN activity.status\_volunteer%type;

PROCEDURE voluntar\_activ(i NUMBER);

FUNCTION ani\_vechime(i NUMBER) RETURN BOOLEAN;

PROCEDURE voluntar\_inactiv(i NUMBER);

PROCEDURE afisare;

END pachet\_info\_voluntari;

/

CREATE OR REPLACE PACKAGE BODY pachet\_info\_voluntari AS

--DECLARATII

TYPE vector IS VARRAY (100) OF VARCHAR2(100);

TYPE matrice IS VARRAY (100) OF vector;

--matrice in care retin lista proiectelor pt fiecare voluntar

mproiecte matrice:=matrice();

n NUMBER:=0;

--matrice cu 2 randuri

--in care retin numele voluntarilor si nr de proiecte

mnume matrice:=matrice();

--FUNCTII/PROCEDURI

--populare matrice cu date

PROCEDURE creare

IS

BEGIN

--calculez nr de voluntari din tabel

SELECT COUNT(volunteer\_id)

INTO n

FROM volunteer;

mnume.extend(n);

mproiecte.extend(n);

--stiu ca id-urile voluntarilor sunt de la 1 la n (16)

FOR i IN 1..n LOOP

mproiecte(i):=vector();

mproiecte(i).extend();

mnume(i):=vector();

mnume(i).extend(2);

--iau numele voluntarilor

SELECT last\_name

INTO mnume(i)(1)

FROM volunteer

WHERE volunteer\_id=i;

--calculez nr de proiecte ca sa evit exceptia value\_error si oprirea programului

SELECT to\_char(COUNT(project\_name))

INTO mnume(i)(2)

FROM works\_on w JOIN projects p ON (w.project\_id=p.project\_id)

WHERE volunteer\_id=i;

IF mnume(i)(2)!='0' THEN

SELECT project\_name BULK COLLECT INTO mproiecte(i)

FROM works\_on w JOIN projects p On (w.project\_id=p.project\_id)

WHERE volunteer\_id=i;

ELSE

mproiecte(i)(1):='Nu are proiecte';

END IF;

END LOOP;

END creare;

--returnez statusul voluntarului cu id-ul i (activ/inactiv/alumn)

FUNCTION verificare\_activ(i NUMBER) RETURN activity.status\_volunteer%type

IS

status activity.status\_volunteer%type;

BEGIN

SELECT status\_volunteer

INTO status

FROM activity a JOIN volunteer v ON (a.activity\_id=v.activity\_id)

WHERE volunteer\_id=i;

RETURN LOWER(status);

END verificare\_activ;

--adaug titlul de membru onorific in nume

PROCEDURE voluntar\_activ(i NUMBER)

IS

BEGIN

IF to\_number(mnume(i)(2))>=2 THEN

mnume(i)(1):=mnume(i)(1) || ' MEMBRU ONORIFIC ';

END IF;

END voluntar\_activ;

--functie care returneaza TRUE daca un voluntar

--are mai mult de 2 ani vechime

FUNCTION ani\_vechime(i NUMBER) RETURN BOOLEAN

IS

ani NUMBER;

BEGIN

SELECT to\_number(to\_char(sysdate, 'yyyy'))-to\_number(to\_char(start\_date, 'yyyy'))

INTO ani

FROM activity a JOIN volunteer v ON (a.activity\_id=v.activity\_id)

WHERE volunteer\_id=i;

IF ani>=2 THEN

RETURN TRUE;

ELSE

RETURN FALSE;

END IF;

END ani\_vechime;

--stergere voluntar i daca nr proiectelor la care a participat este <= 1

PROCEDURE voluntar\_inactiv(i NUMBER)

IS

BEGIN

IF to\_number(mnume(i)(2))<=1 THEN

DELETE FROM volunteer

WHERE volunteer\_id=i;

mnume(i)(1):= mnume(i)(1) ||' a fost sters ';

END IF;

END voluntar\_inactiv;

--procedura de afisare si apelare functii

PROCEDURE afisare

IS

BEGIN

creare();

FOR i IN 1..n LOOP

--un voluntar poate fi activ/inactiv/alumn/presedinte

--modificarile au loc doar pt cei activi/inactivi

mnume(i)(1):='Voluntarul '|| verificare\_activ(i)|| ' '|| mnume(i)(1);

IF verificare\_activ(i)='activ' THEN

--voluntar activ

--daca au participat la cel putin 2 proiecte

--le adaug la afisare titlul de membru onorific

voluntar\_activ(i);

ELSIF verificare\_activ(i)='inactiv' AND ani\_vechime(i)=TRUE THEN

--voluntar inactiv

--daca are cel putin 2 ani vechime

--si maxim 1 proiect la care a participat

--va fi sters din baza de date

--si afisat un mesaj corespunzator

voluntar\_inactiv(i);

END IF;

END LOOP;

FOR i IN 1..n LOOP

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

DBMS\_OUTPUT.PUT\_LINE(mnume(i)(1));

DBMS\_OUTPUT.PUT\_LINE('Proiectele la care a participat: ');

FOR j IN mproiecte(i).first..mproiecte(i).last LOOP

Dbms\_Output.Put\_Line(mproiecte(i)(j));

END LOOP;

END LOOP;

END afisare;

END pachet\_info\_voluntari;

/

BEGIN

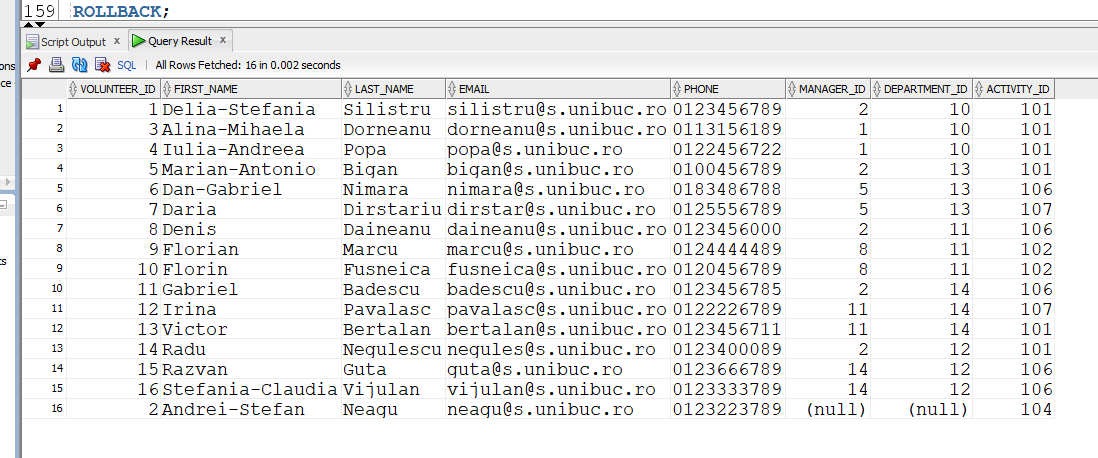
pachet\_info\_voluntari.afisare();

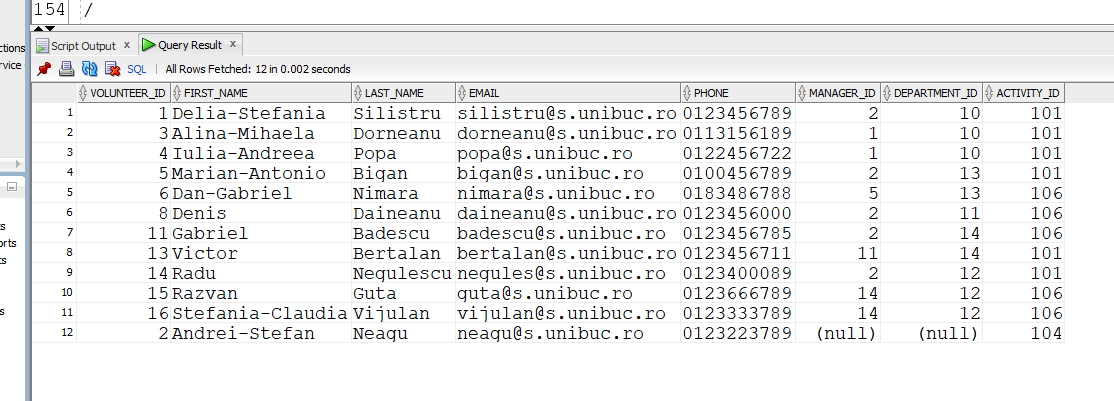
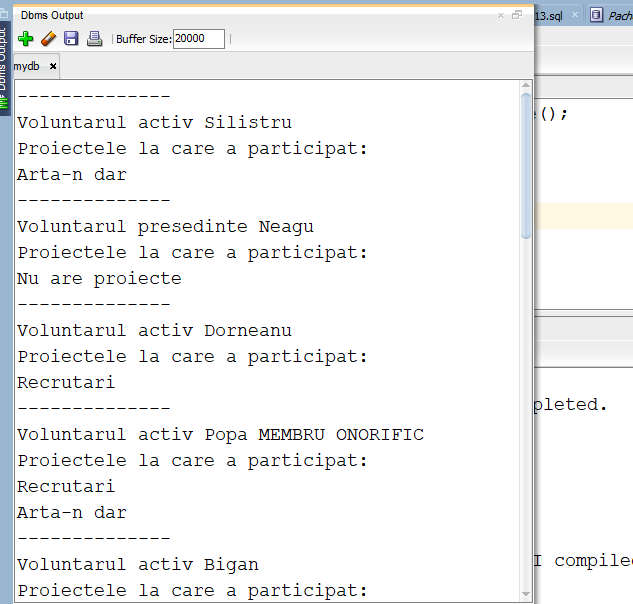
END;

/

ROLLBACK;

/

**BEFORE:**

**AFTER:**

