Proiect SGBD

Agușoaei Alexandru-Gabriel

Seria: 23

Grupa 233

An universitar 2024-2025

**Baza de date a unei platforme de socializare**

Contents

[1. Prezentare baza de date 4](#_Toc187416887)

[2. Diagrama ER 5](#_Toc187416888)

[3. Diagrama Conceptuală 6](#_Toc187416889)

[4. Implementarea în Oracle a diagramei conceptuale 7](#_Toc187416890)

[5. Inserari de date in tabele 13](#_Toc187416891)

[6. Subprogram stocat independentcare să utilizeze toate cele 3 tipuri de colecții studiate 29](#_Toc187416892)

[7. Subprogram stocat independent care să utilizeze 2 tipuri de cursoare 31](#_Toc187416893)

[8. Subprogram stocat independent de tip funcție care sa utilizeze 3 tabele într-o singură comandă 34](#_Toc187416894)

[9. Subprogram stocat independent de tip procedură cu 2 parametrii care să utilizeze 5 tabele într-o comandă 36](#_Toc187416895)

[10. Trigger de tip LMD la nivel de comandă 39](#_Toc187416896)

[11. Trigger de tip LMD la nivel de linie 41](#_Toc187416897)

[12. Trigger de tip LDD 43](#_Toc187416898)

[13. Pachet care să includă 2 tipuri de date complexe, 2 funcții și 2 proceduri 44](#_Toc187416899)

Cerințe

1. Prezentați pe scurt baza de date (utilitatea ei).
2. Realizați **diagrama entitate-relație** (ERD): entitățile, relațiile și atributele trebuie definite în **limba română** (vezi curs SGBD, model de diagrama entitate-relație; nu se va accepta alt format).
3. Pornind de la diagrama entitate-relație realizați **diagrama conceptuală** a modelului propus, integrând toate atributele necesare: entitățile, relațiile și atributele trebuie definite în **limba română**.
4. Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, adăugând toate constrângerile de integritate necesare (chei primare, cheile externe etc).
5. Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru fiecare tabelă asociativă).
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.
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 subprogramul.
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 create. Tratați toate excepțiile care pot apărea, incluzând excepțiile predefinite NO\_DATA\_FOUND și TOO\_MANY\_ROWS. Apelați subprogramul astfel încât să evidențiați **toate** cazurile tratate.
9. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un **subprogram stocat independent de tip procedură** care să aibă minim 2 parametri și să utilizeze **într-o singură comandă SQL** 5 dintre tabelele create. Definiți minim 2 excepții proprii, altele decât cele predefinite la nivel de sistem. Apelați subprogramul astfel încât să evidențiați **toate** cazurile definite și tratate.

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

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

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

13. Formulați în limbaj natural o problemă pe care să o rezolvați folosind 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).

1. Prezentare baza de date

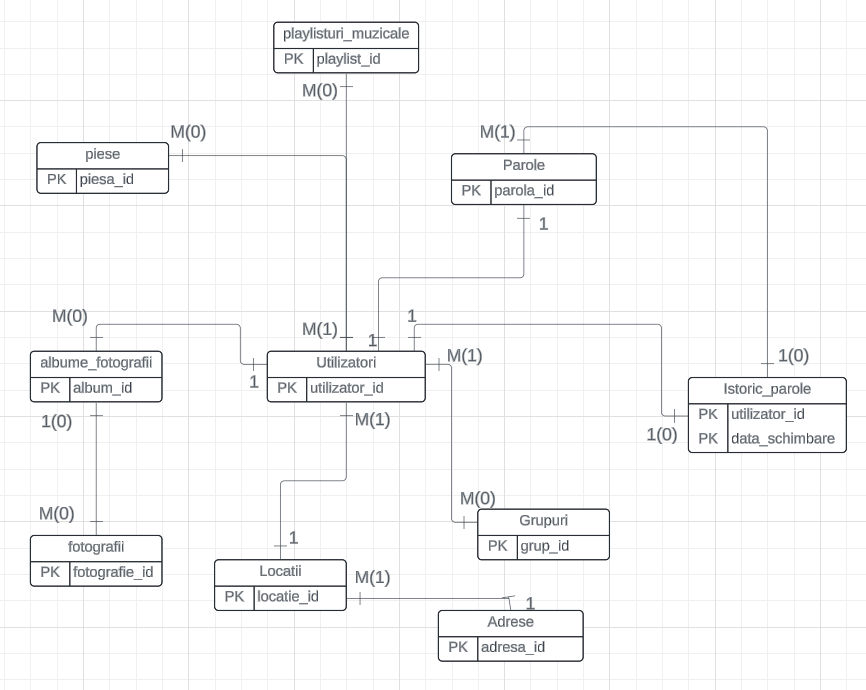
În acest proiect voi ilustra cum arată o bază de date a unei platforme de socializare. În era digitală actuală, platformele de socializare au devenit componente esențiale ale vieții cotidiene, conectând milioane de utilizatori din întreaga lume. Aceste platforme permit oamenilor să interacționeze, să împărtășească idei, să formeze comunități și să creeze conținut, transformând modul în care comunicăm și ne organizăm societatea. În spatele funcționării eficiente a acestor platforme complexe se află un element tehnologic critic: baza de date. O bază de date bine proiectată și gestionată este fundamentul pe care se sprijină orice platformă de socializare. Aceasta joacă un rol vital în stocarea, gestionarea și accesarea informațiilor utilizatorilor, facilitând o varietate de funcționalități esențiale pentru experiența utilizatorilor.

Versiune : 19.2.1

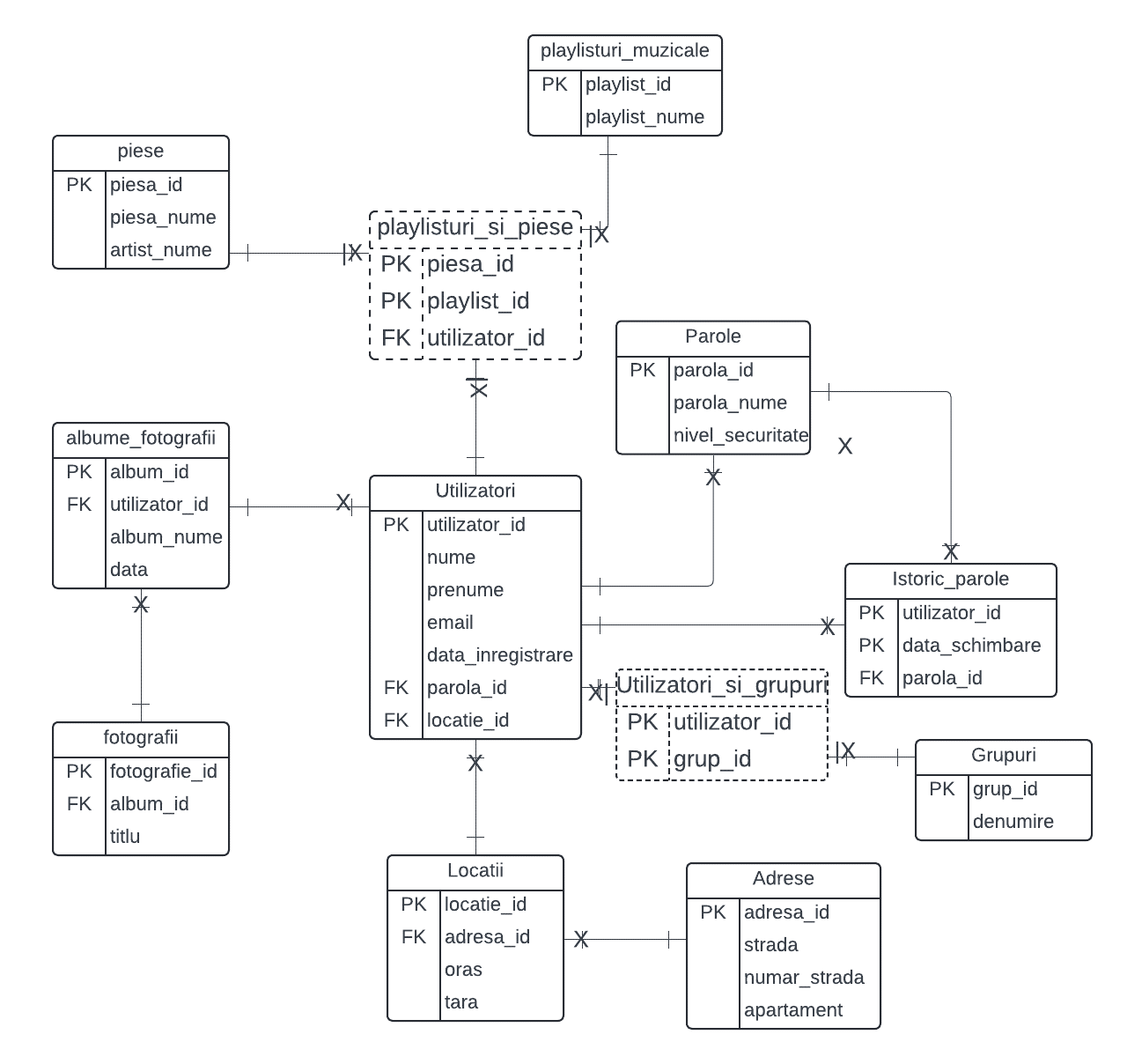
Configurație software: Windows 10, nu am folosit masina virtuala

RAM Utilizat: 1138MB

2. Diagrama ER



# 3. Diagrama Conceptuală



# 4. Implementarea în Oracle a diagramei conceptuale

-- Crearea tabelei utilizatori

CREATE TABLE utilizatori

(

utilizator\_id number(5) constraint pk\_utilizator primary key,

nume varchar(100) constraint nume\_utilizator not null,

prenume varchar(100) constraint prenume\_utilizator not null,

email varchar(100) constraint email\_utilizator not null,

parola\_id number(5),

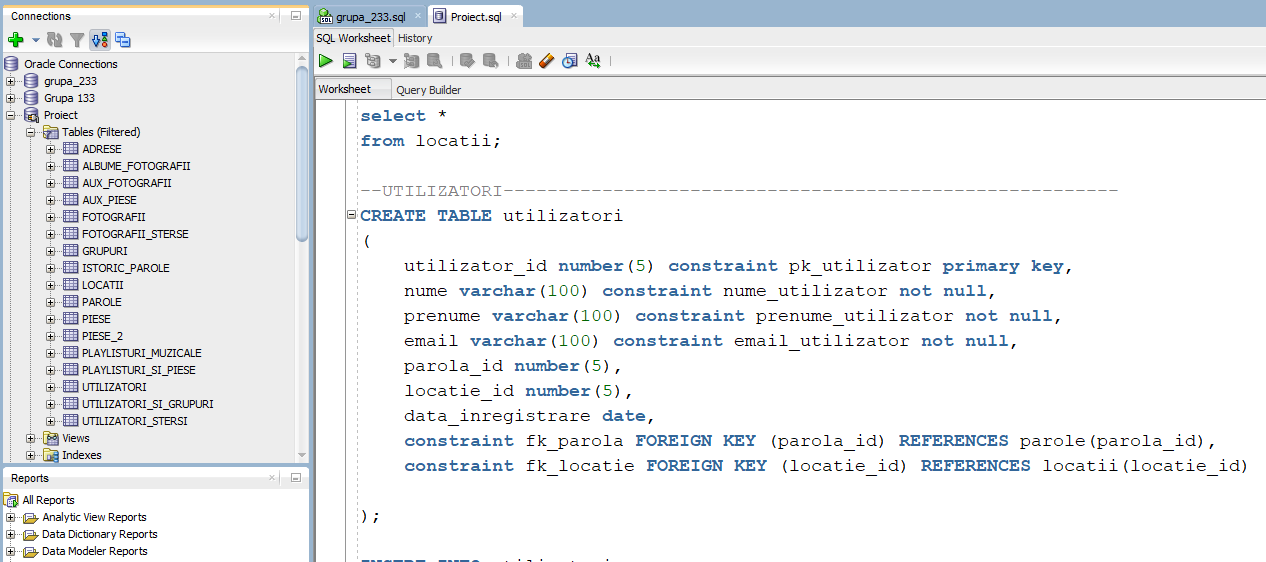
locatie\_id number(5),

data\_inregistrare date,

constraint fk\_parola FOREIGN KEY (parola\_id) REFERENCES parole(parola\_id),

constraint fk\_locatie FOREIGN KEY (locatie\_id) REFERENCES locatii(locatie\_id)

);



-- Crearea tabelei parole

CREATE TABLE parole

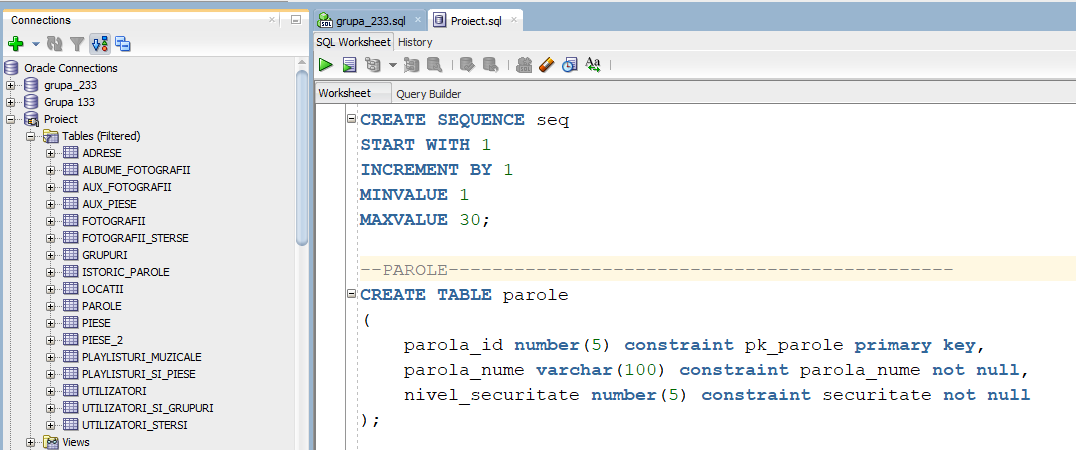
(

parola\_id number(5) constraint pk\_parole primary key,

parola\_nume varchar(100) constraint parola\_nume not null,

nivel\_securitate number(5) constraint securitate not null

);



-- Crearea tabelei adrese

CREATE TABLE adrese

(

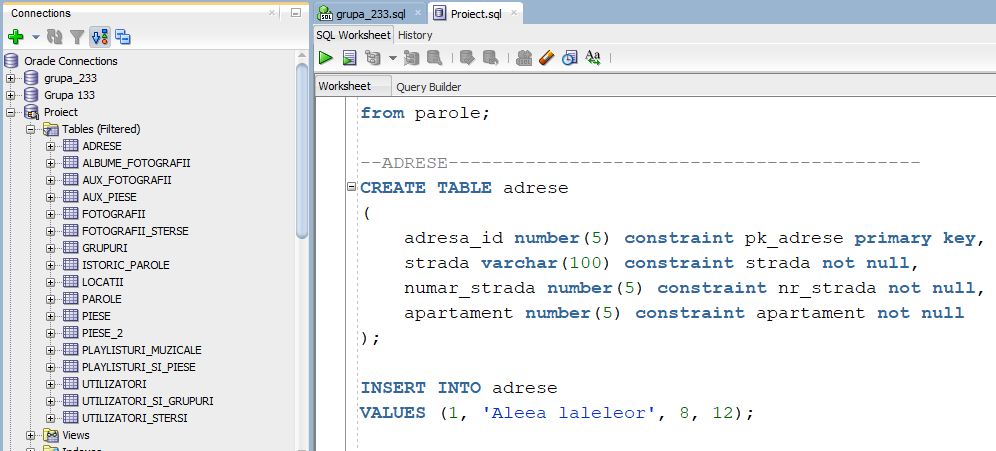
adresa\_id number(5) constraint pk\_adrese primary key,

strada varchar(100) constraint strada not null,

numar\_strada number(5) constraint nr\_strada not null,

apartament number(5) constraint apartament not null

);



-- Crearea tabelei locatii

CREATE TABLE locatii

(

locatie\_id number(5) constraint pk\_locatii primary key,

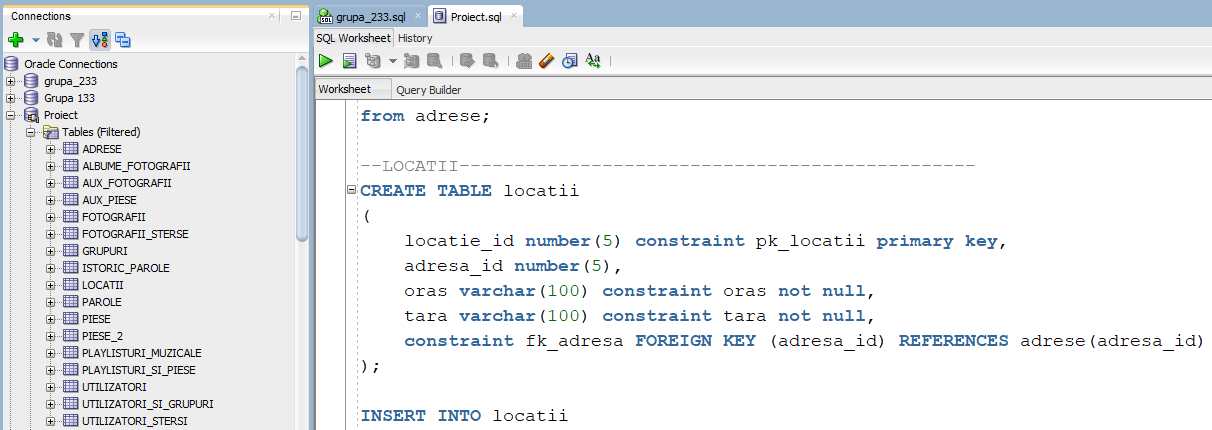
adresa\_id number(5),

oras varchar(100) constraint oras not null,

tara varchar(100) constraint tara not null,

constraint fk\_adresa FOREIGN KEY (adresa\_id) REFERENCES adrese(adresa\_id)

);



-- Crearea tabelei grupuri

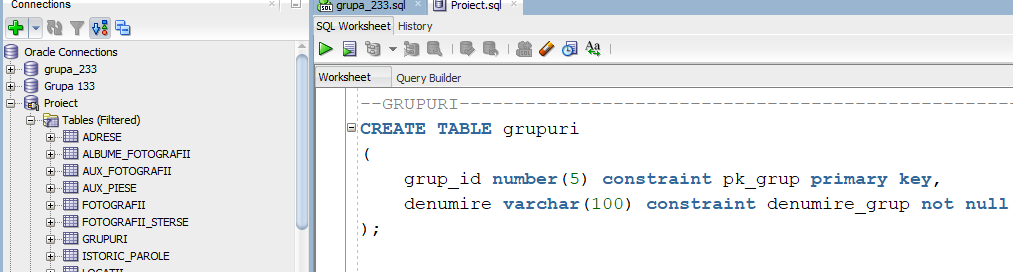
CREATE TABLE grupuri

(

grup\_id number(5) constraint pk\_grup primary key,

denumire varchar(100) constraint denumire\_grup not null

);



-- Crearea tabelei utilizatori\_si\_grupuri

CREATE TABLE utilizatori\_si\_grupuri

(

utilizator\_id number(5),

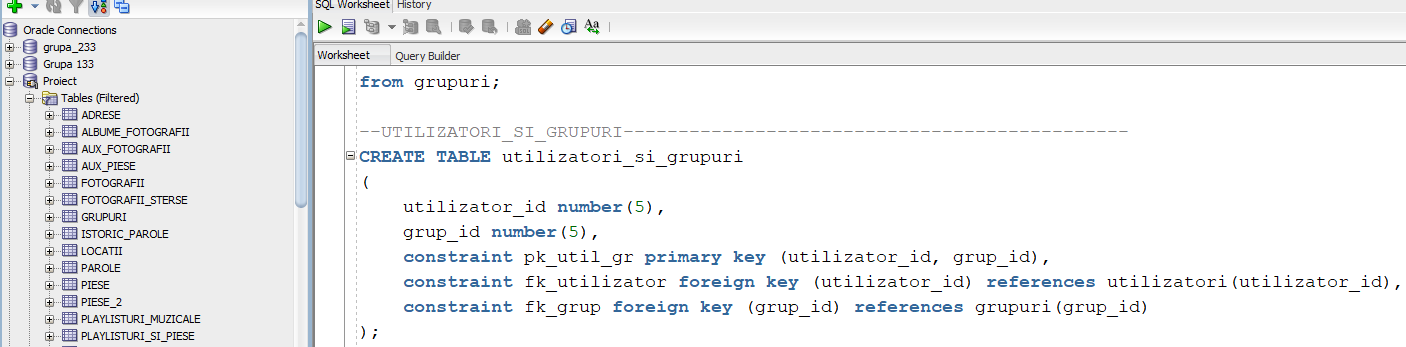
grup\_id number(5),

constraint pk\_util\_gr primary key (utilizator\_id, grup\_id),

constraint fk\_utilizator foreign key (utilizator\_id) references utilizatori(utilizator\_id),

constraint fk\_grup foreign key (grup\_id) references grupuri(grup\_id)

);



-- Crearea tabelei fotografii

CREATE TABLE fotografii

(

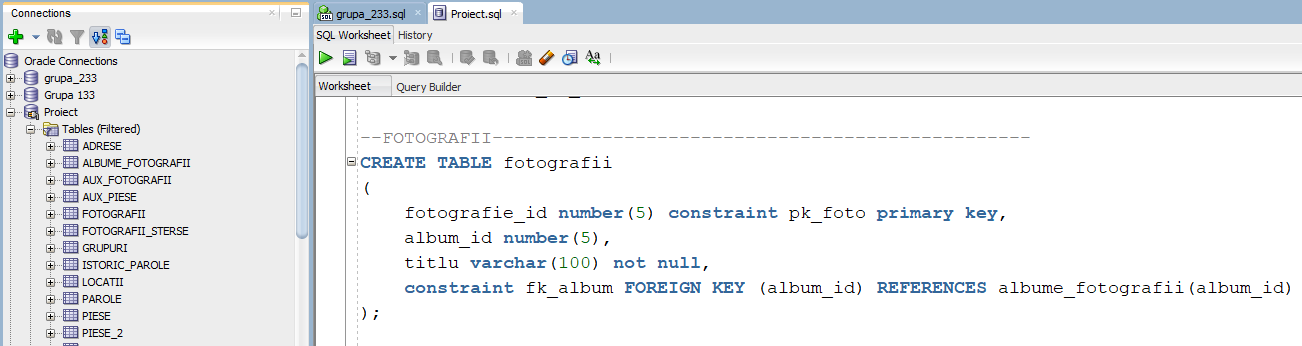
fotografie\_id number(5) constraint pk\_foto primary key,

album\_id number(5),

titlu varchar(100) not null,

constraint fk\_album FOREIGN KEY (album\_id) REFERENCES albume\_fotografii(album\_id)

);



-- Crearea tabelei piese

CREATE TABLE piese

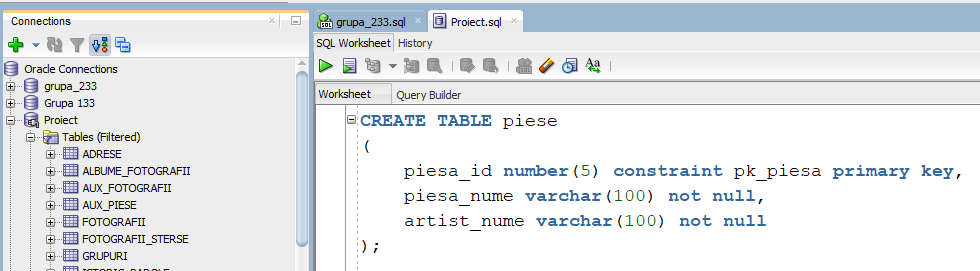
(

piesa\_id number(5) constraint pk\_piesa primary key,

piesa\_nume varchar(100) not null,

artist\_nume varchar(100) not null

);



-- Crearea tabelei playlisturi\_muzicale

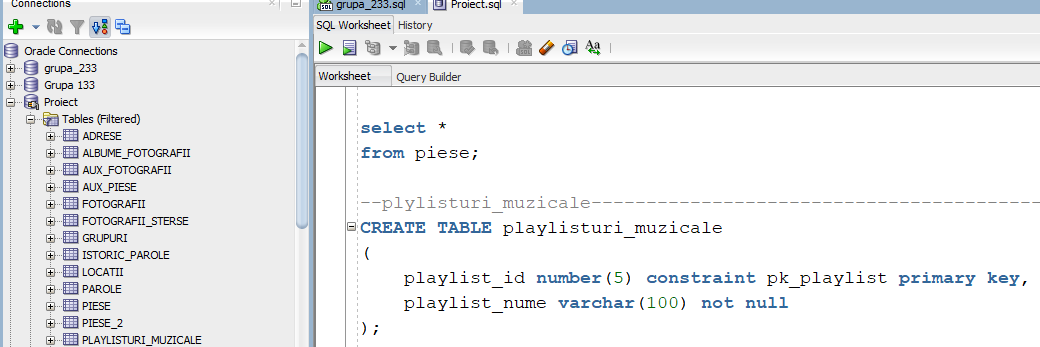
CREATE TABLE playlisturi\_muzicale

(

playlist\_id number(5) constraint pk\_playlist primary key,

playlist\_nume varchar(100) not null

);



-- Crearea tabelei playlisturi\_si\_piese

CREATE TABLE playlisturi\_si\_piese

(

piesa\_id number(5),

playlist\_id number(5),

utilizator\_id number(5),

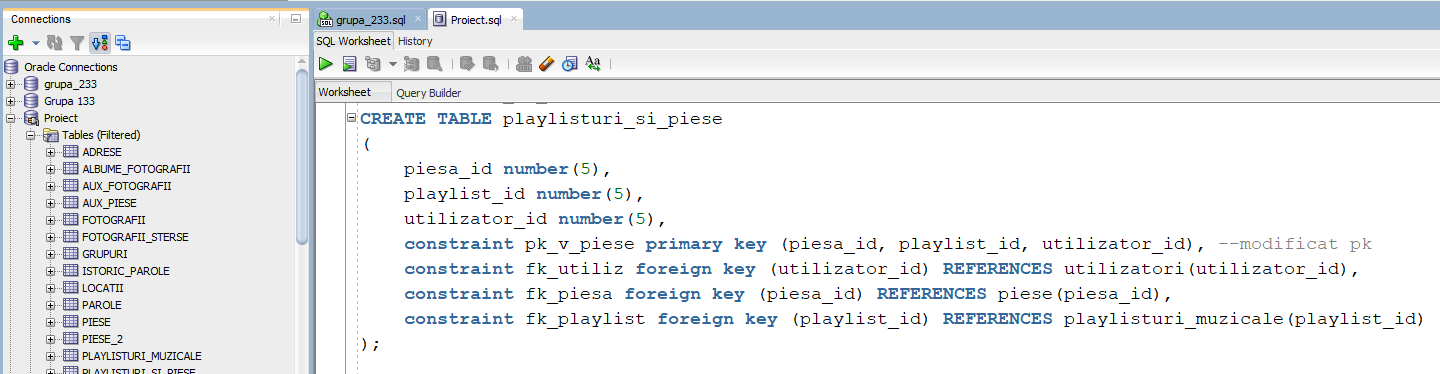
constraint pk\_v\_piese primary key (piesa\_id, playlist\_id, utilizator\_id), --modificat pk

constraint fk\_utiliz foreign key (utilizator\_id) REFERENCES utilizatori(utilizator\_id),

constraint fk\_piesa foreign key (piesa\_id) REFERENCES piese(piesa\_id),

constraint fk\_playlist foreign key (playlist\_id) REFERENCES playlisturi\_muzicale(playlist\_id)

);



-- Crearea tabelei istoric\_parole

CREATE TABLE istoric\_parole

(

utilizator\_id number(5),

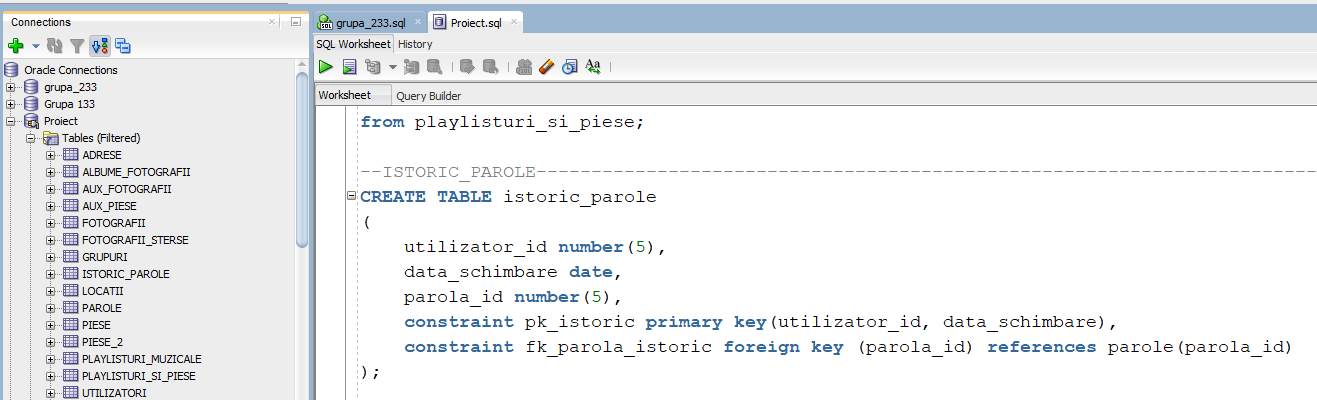
data\_schimbare date,

parola\_id number(5),

constraint pk\_istoric primary key(utilizator\_id, data\_schimbare),

constraint fk\_parola\_istoric foreign key (parola\_id) references parole(parola\_id)

);



# 5. Inserari de date in tabele

*Utilizatori*



INSERT INTO utilizatori

VALUES (seq.NEXTVAL, 'Popescu', 'Andrei', 'popescu\_andrei@gmail.com', 1, 1, TO\_DATE('04/12/2023', 'DD/MM/YYYY'));

INSERT INTO utilizatori

VALUES (seq.NEXTVAL, 'Dimitrie', 'Roxana', 'dimitrie\_roxx@gmail.com', 3, 4, TO\_DATE('12/06/2023', 'DD/MM/YYYY'));

INSERT INTO utilizatori

VALUES (seq.NEXTVAL, 'Ionescu', 'Maria', 'ionescu\_maria@gmail.com', 2, 2, TO\_DATE('05/11/2023', 'DD/MM/YYYY'));

INSERT INTO utilizatori

VALUES (seq.NEXTVAL, 'Georgescu', 'Mihai', 'georgescu\_mihai@gmail.com', 5, 3, TO\_DATE('06/10/2023', 'DD/MM/YYYY'));

INSERT INTO utilizatori

VALUES (seq.NEXTVAL, 'Vasilescu', 'Elena', 'vasilescu\_elena@gmail.com', 4, 3, TO\_DATE('22/12/2023', 'DD/MM/YYYY'));

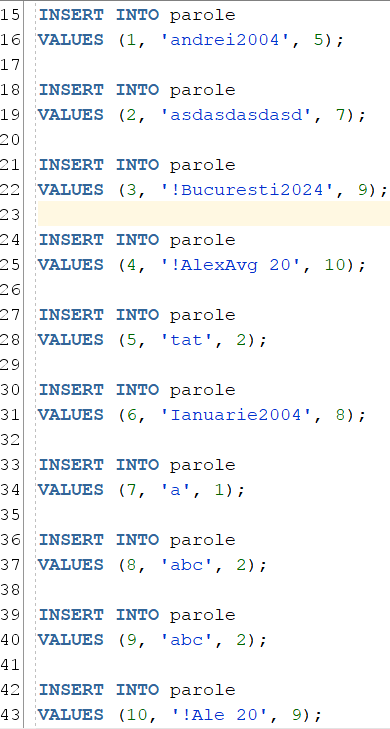
INSERT INTO utilizatori

VALUES (seq.NEXTVAL, 'Ionescu', 'Elena', 'ionescu\_elena@gmail.com', 7, 2, TO\_DATE('05/11/2023', 'DD/MM/YYYY'));

INSERT INTO utilizatori

VALUES (31, 'Mihailovici', 'Andrei', 'miha\_andrei@gmail.com', 11, 6, TO\_DATE('02/04/2023', 'DD/MM/YYYY'));

*Parole*



INSERT INTO parole

VALUES (1, 'andrei2004', 5);

INSERT INTO parole

VALUES (2, 'asdasdasdasd', 7);

INSERT INTO parole

VALUES (3, '!Bucuresti2024', 9);

INSERT INTO parole

VALUES (4, '!AlexAvg 20', 10);

INSERT INTO parole

VALUES (5, 'tat', 2);

INSERT INTO parole

VALUES (6, 'Ianuarie2004', 8);

INSERT INTO parole

VALUES (7, 'a', 1);

INSERT INTO parole

VALUES (8, 'abc', 2);

INSERT INTO parole

VALUES (9, 'abc', 2);

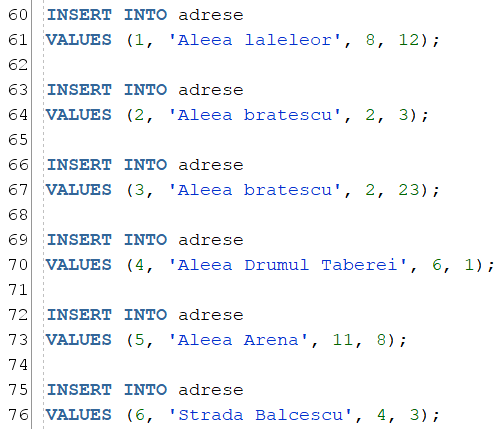
INSERT INTO parole

VALUES (10, '!Ale 20', 9);

INSERT INTO parole

VALUES (11, '!AavFe', 8);

*Adrese*



INSERT INTO adrese

VALUES (1, 'Aleea laleleor', 8, 12);

INSERT INTO adrese

VALUES (2, 'Aleea bratescu', 2, 3);

INSERT INTO adrese

VALUES (3, 'Aleea bratescu', 2, 23);

INSERT INTO adrese

VALUES (4, 'Aleea Drumul Taberei', 6, 1);

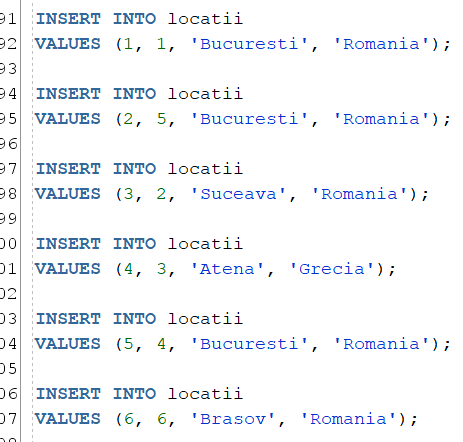
INSERT INTO adrese

VALUES (5, 'Aleea Arena', 11, 8);

INSERT INTO adrese

VALUES (6, 'Strada Balcescu', 4, 3);

*Locatii*



INSERT INTO locatii

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

INSERT INTO locatii

VALUES (2, 5, 'Bucuresti', 'Romania');

INSERT INTO locatii

VALUES (3, 2, 'Suceava', 'Romania');

INSERT INTO locatii

VALUES (4, 3, 'Atena', 'Grecia');

INSERT INTO locatii

VALUES (5, 4, 'Bucuresti', 'Romania');

INSERT INTO locatii

VALUES (6, 6, 'Brasov', 'Romania');

*Grupuri*

INSERT INTO grupuri

VALUES (100, 'Scoala');

INSERT INTO grupuri

VALUES (101, 'La munca');

INSERT INTO grupuri

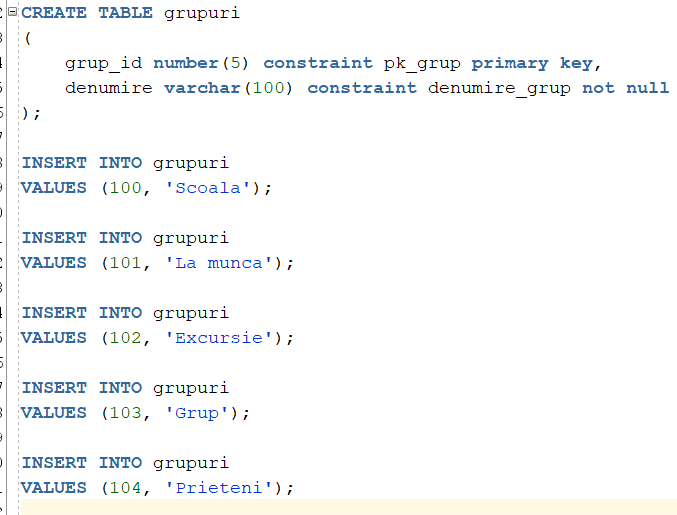
VALUES (102, 'Excursie');

INSERT INTO grupuri

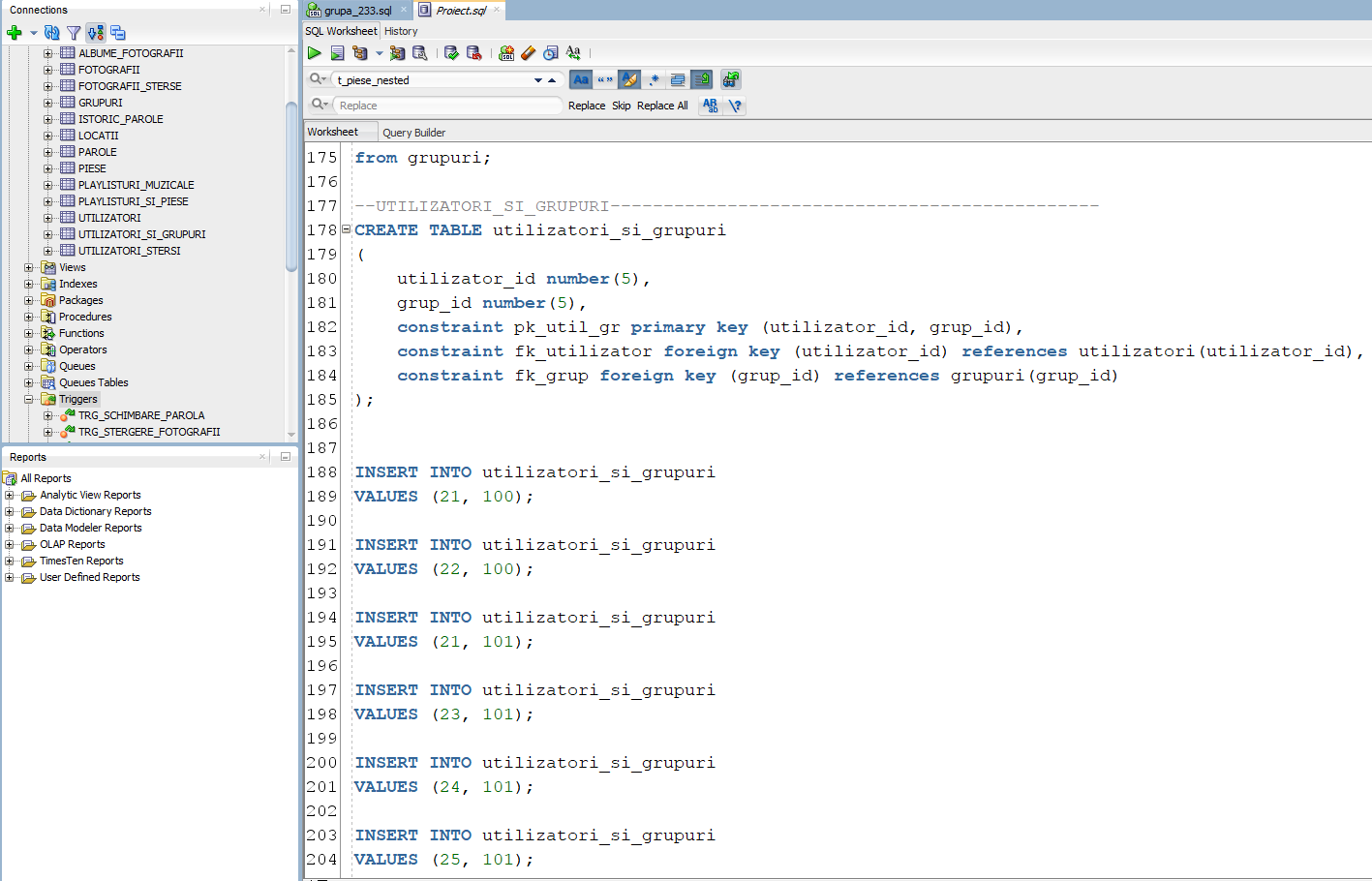
VALUES (103, 'Grup');

INSERT INTO grupuri

VALUES (104, 'Prieteni');



*Utilizatori\_si\_grupuri*

INSERT INTO utilizatori\_si\_grupuri

VALUES (21, 100);

INSERT INTO utilizatori\_si\_grupuri

VALUES (22, 100);

INSERT INTO utilizatori\_si\_grupuri

VALUES (21, 101);

INSERT INTO utilizatori\_si\_grupuri

VALUES (23, 101);

INSERT INTO utilizatori\_si\_grupuri

VALUES (24, 101);

INSERT INTO utilizatori\_si\_grupuri

VALUES (25, 101);

INSERT INTO utilizatori\_si\_grupuri

VALUES (26, 101);

INSERT INTO utilizatori\_si\_grupuri

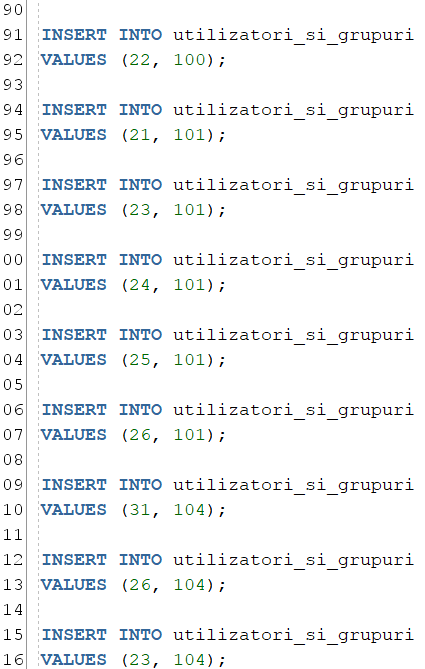
VALUES (31, 104);

INSERT INTO utilizatori\_si\_grupuri

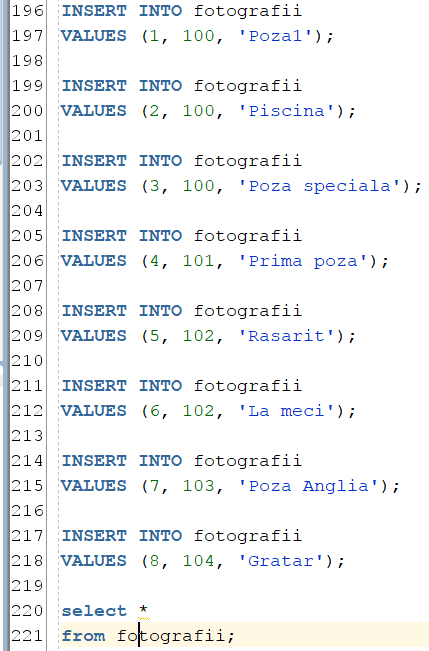
VALUES (26, 104);

INSERT INTO utilizatori\_si\_grupuri

VALUES (23, 104);



*Fotografii*



INSERT INTO fotografii

VALUES (1, 100, 'Poza1');

INSERT INTO fotografii

VALUES (2, 100, 'Piscina');

INSERT INTO fotografii

VALUES (3, 100, 'Poza speciala');

INSERT INTO fotografii

VALUES (4, 101, 'Prima poza');

INSERT INTO fotografii

VALUES (5, 102, 'Rasarit');

INSERT INTO fotografii

VALUES (6, 102, 'La meci');

INSERT INTO fotografii

VALUES (7, 103, 'Poza Anglia');

INSERT INTO fotografii

VALUES (8, 104, 'Gratar')

*Albume fotografii*



CREATE TABLE albume\_fotografii

(

album\_id number(5) constraint pk\_album primary key,

utilizator\_id number(5),

album\_nume varchar(100) constraint nume\_album not null,

data\_album date,

constraint fk\_util FOREIGN KEY (utilizator\_id) REFERENCES utilizatori(utilizator\_id)

);

INSERT INTO albume\_fotografii

VALUES (100, 21, 'La mare', TO\_DATE('25/12/2023', 'DD/MM/YYYY'));

INSERT INTO albume\_fotografii

VALUES (101, 21, 'Ziua mea', TO\_DATE('01/02/2024', 'DD/MM/YYYY'));

INSERT INTO albume\_fotografii

VALUES (102, 23, 'Scoala', TO\_DATE('29/11/2023', 'DD/MM/YYYY'));

INSERT INTO albume\_fotografii

VALUES (103, 24, 'Facultate', TO\_DATE('15/08/2023', 'DD/MM/YYYY'));

INSERT INTO albume\_fotografii

VALUES (104, 24, 'La mare', TO\_DATE('02/03/2024', 'DD/MM/YYYY'));

*Piese*

INSERT INTO piese (piesa\_id, piesa\_nume, artist\_nume)

VALUES (1, 'Shape of You', 'Ed Sheeran');

INSERT INTO piese (piesa\_id, piesa\_nume, artist\_nume)

VALUES (2, 'Blinding Lights', 'The Weeknd');

INSERT INTO piese (piesa\_id, piesa\_nume, artist\_nume)

VALUES (3, 'Levitating', 'Dua Lipa');

INSERT INTO piese (piesa\_id, piesa\_nume, artist\_nume)

VALUES (4, 'Bad Guy', 'Billie Eilish');

INSERT INTO piese (piesa\_id, piesa\_nume, artist\_nume)

VALUES (5, 'Rockstar', 'Post Malone');

INSERT INTO piese (piesa\_id, piesa\_nume, artist\_nume)

VALUES (6, 'Old Town Road', 'Lil Nas X');

INSERT INTO piese (piesa\_id, piesa\_nume, artist\_nume)

VALUES (7, 'Senorita', 'Camila Cabello');

INSERT INTO piese (piesa\_id, piesa\_nume, artist\_nume)

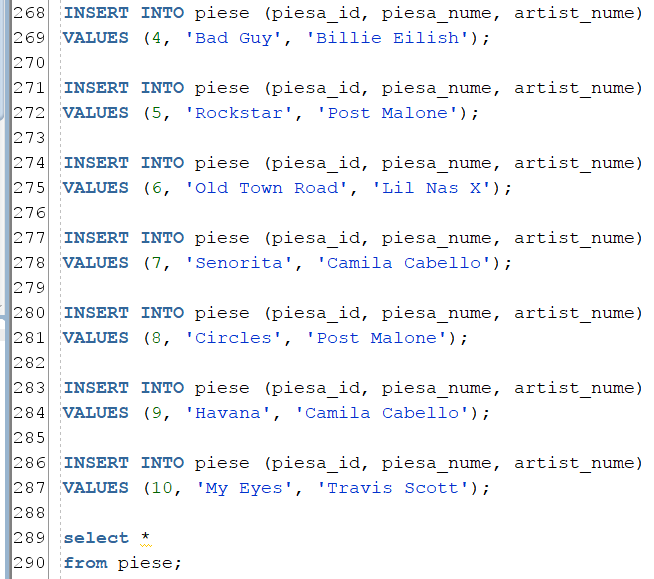
VALUES (8, 'Circles', 'Post Malone');

INSERT INTO piese (piesa\_id, piesa\_nume, artist\_nume)

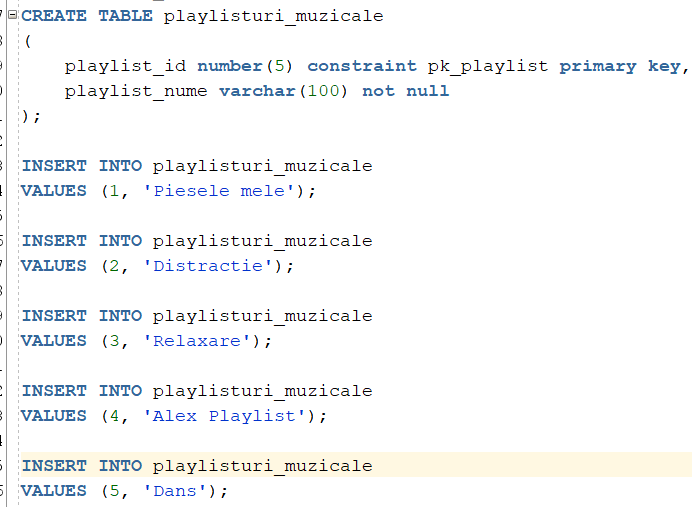
VALUES (9, 'Havana', 'Camila Cabello');

INSERT INTO piese (piesa\_id, piesa\_nume, artist\_nume)

VALUES (10, 'My Eyes', 'Travis Scott');



*Playlisturi\_muzicale*



INSERT INTO playlisturi\_muzicale

VALUES (1, 'Piesele mele');

INSERT INTO playlisturi\_muzicale

VALUES (2, 'Distractie');

INSERT INTO playlisturi\_muzicale

VALUES (3, 'Relaxare');

INSERT INTO playlisturi\_muzicale

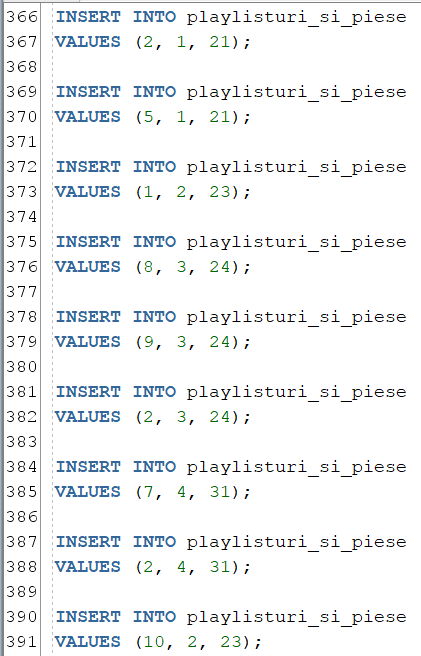
VALUES (4, 'Alex Playlist');

INSERT INTO playlisturi\_muzicale

VALUES (5, 'Dans');

*Playlisturi\_si\_piese*

INSERT INTO playlisturi\_si\_piese

VALUES (1, 1, 21);

INSERT INTO playlisturi\_si\_piese

VALUES (2, 1, 21);

INSERT INTO playlisturi\_si\_piese

VALUES (5, 1, 21);

INSERT INTO playlisturi\_si\_piese

VALUES (1, 2, 23);

INSERT INTO playlisturi\_si\_piese

VALUES (8, 3, 24);

INSERT INTO playlisturi\_si\_piese

VALUES (9, 3, 24);

INSERT INTO playlisturi\_si\_piese

VALUES (2, 3, 24);

INSERT INTO playlisturi\_si\_piese

VALUES (7, 4, 31);

INSERT INTO playlisturi\_si\_piese

VALUES (2, 4, 31);

INSERT INTO playlisturi\_si\_piese

VALUES (10, 2, 23);

*Istoric\_parole*



INSERT INTO istoric\_parole

VALUES (22, TO\_DATE('27/12/2023', 'DD/MM/YYYY'), 8);

INSERT INTO istoric\_parole

VALUES (22, TO\_DATE('01/01/2024', 'DD/MM/YYYY'), 9);

INSERT INTO istoric\_parole

VALUES (24, TO\_DATE('30/12/2023', 'DD/MM/YYYY'), 10);

INSERT INTO istoric\_parole

VALUES (31, TO\_DATE('01/02/2024', 'DD/MM/YYYY'), 1);

INSERT INTO istoric\_parole

VALUES (31, TO\_DATE('04/02/2023', 'DD/MM/YYYY'), 3);

# 6. Subprogram stocat independentcare să utilizeze toate cele 3 tipuri de colecții studiate

Cerință: Să se afișeze piesele, playlisturile muzicale si grupurile utilizatorului cu id-ul dat ca parametru. (3 tipuri de colectii)

create or replace procedure ex6(

utilizator\_id\_input utilizatori.utilizator\_id%type -- id-ul utilizatorului

)

is

-- 1. colectie de tip tablou imbricat pentru piesele utilizatorului

type colectie\_piese is table of piese.piesa\_nume%type;

v\_piese colectie\_piese;

-- 2. colectie de tip varray pentru playlisturile utilizatorului

type colectie\_playlisturi is varray(10) of playlisturi\_muzicale.playlist\_nume%type;

v\_playlisturi colectie\_playlisturi;

-- 3. colectie de tip tablou indexat pentru grupurile utilizatorului

type colectie\_grupuri is table of grupuri.denumire%type index by pls\_integer;

v\_grupuri colectie\_grupuri;

-- numele utilizatorului

nume\_utilizator varchar2(20);

prenume\_utilizator varchar2(20);

begin

-- obtinem numele utilizatorului

select u.nume, u.prenume

into nume\_utilizator, prenume\_utilizator

from utilizatori u

where u.utilizator\_id = utilizator\_id\_input;

-- afisam numele utilizatorului

dbms\_output.put\_line('Numele utilizatorului: ' || nume\_utilizator || ' ' || prenume\_utilizator);

-- 1. selectam piesele din playlisturile utilizatorului

select distinct p.piesa\_nume

bulk collect into v\_piese

from piese p

join playlisturi\_si\_piese pp on p.piesa\_id = pp.piesa\_id

where pp.utilizator\_id = utilizator\_id\_input;

-- 2. selectam playlisturile utilizatorului

select distinct pm.playlist\_nume

bulk collect into v\_playlisturi

from playlisturi\_muzicale pm

join playlisturi\_si\_piese pp on pm.playlist\_id = pp.playlist\_id

where pp.utilizator\_id = utilizator\_id\_input;

-- 3. selectam grupurile utilizatorului

select g.denumire

bulk collect into v\_grupuri

from grupuri g

join utilizatori\_si\_grupuri ug on g.grup\_id = ug.grup\_id

where ug.utilizator\_id = utilizator\_id\_input;

dbms\_output.put\_line('Piese din playlisturile utilizatorului:');

for i in 1..v\_piese.count loop

dbms\_output.put\_line('- ' || v\_piese(i));

end loop;

dbms\_output.put\_line('Playlisturile utilizatorului:');

for i in 1..v\_playlisturi.count loop

dbms\_output.put\_line('- ' || v\_playlisturi(i));

end loop;

-- afisam grupurile din care face parte utilizatorul

dbms\_output.put\_line('Grupurile din care face parte utilizatorul:');

for i in 1..v\_grupuri.count loop

dbms\_output.put\_line('- ' || v\_grupuri(i));

end loop;

end ex6;

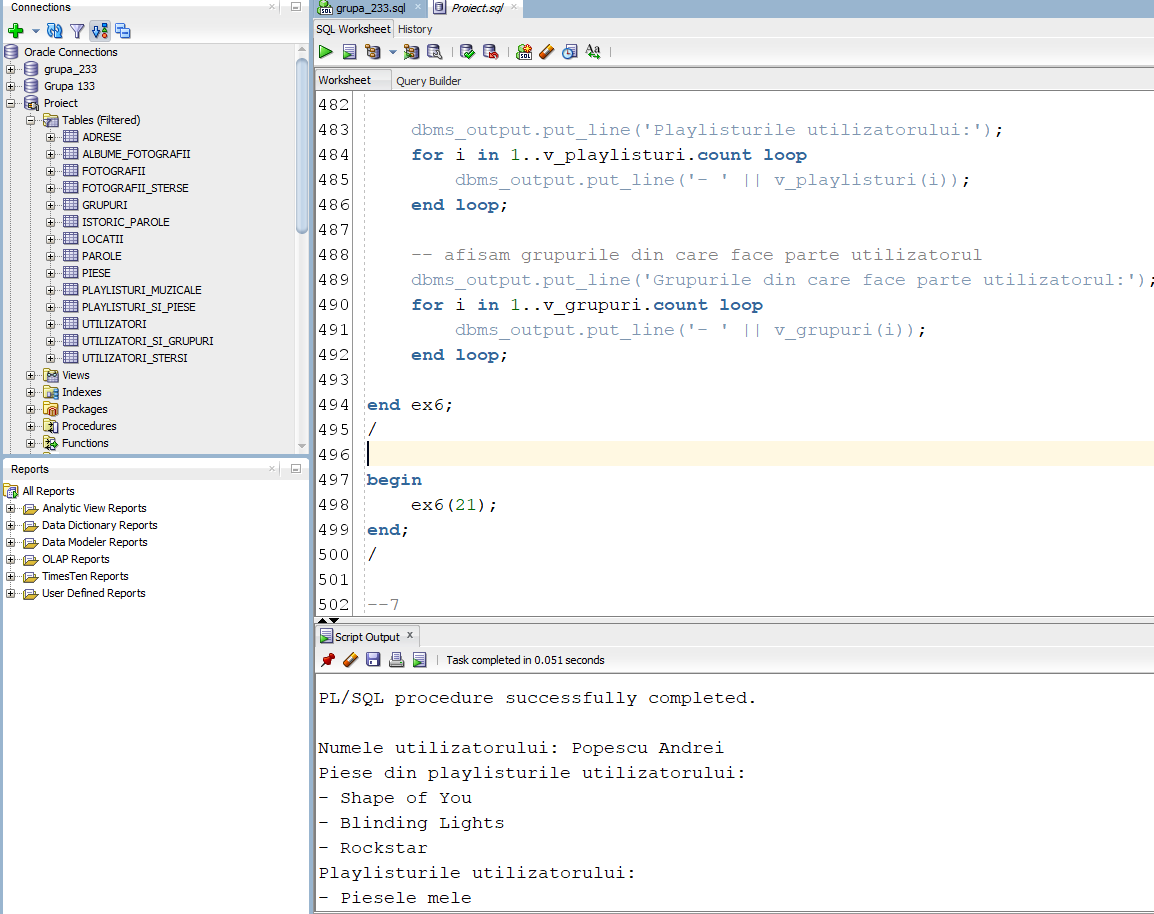
/

begin

ex6(21);

end;

/



# 7. Subprogram stocat independent care să utilizeze 2 tipuri de cursoare

Cerință: Să se afișeze albumele de fotografii împreună cu fotografiile asociate utilizatorului cu id-ul dat ca parametru. (2 tipuri de cursoare)

create or replace procedure ex7(

p\_utilizator\_id utilizatori.utilizator\_id%type

) is

-- variabile pentru utilizator

v\_nume utilizatori.nume%type;

v\_prenume utilizatori.prenume%type;

v\_email utilizatori.email%type;

-- cursor explicit pentru albume

cursor albume\_cursor is

select album\_id, album\_nume, data\_album

from albume\_fotografii

where utilizator\_id = p\_utilizator\_id;

-- variabile pentru albume

v\_album\_id albume\_fotografii.album\_id%type;

v\_album\_nume albume\_fotografii.album\_nume%type;

v\_data\_album albume\_fotografii.data\_album%type;

-- cursor parametrizat pentru fotografii

cursor fotografii\_cursor(p\_album\_id fotografii.album\_id%type) is

select titlu

from fotografii

where album\_id = p\_album\_id;

-- variabila pentru fotografii

v\_titlu fotografii.titlu%type;

begin

begin

select nume, prenume, email

into v\_nume, v\_prenume, v\_email

from utilizatori

where utilizator\_id = p\_utilizator\_id;

dbms\_output.put\_line('Utilizator: ' || v\_nume || ' ' || v\_prenume || ' - ' || v\_email);

exception

when no\_data\_found then

dbms\_output.put\_line('Utilizatorul cu id-ul ' || p\_utilizator\_id || ' nu exista.');

return;

end;

-- procesam albumele utilizatorului folosind cursorul explicit

open albume\_cursor;

loop

fetch albume\_cursor into v\_album\_id, v\_album\_nume, v\_data\_album;

exit when albume\_cursor%notfound;

dbms\_output.put\_line('Album: ' || v\_album\_nume || ' (data: ' || to\_char(v\_data\_album, 'dd/mm/yyyy') || ')');

-- procesam fotografiile asociate albumului curent

open fotografii\_cursor(v\_album\_id);

loop

fetch fotografii\_cursor into v\_titlu;

exit when fotografii\_cursor%notfound;

dbms\_output.put\_line('- fotografie: ' || v\_titlu);

end loop;

close fotografii\_cursor;

end loop;

close albume\_cursor;

end;

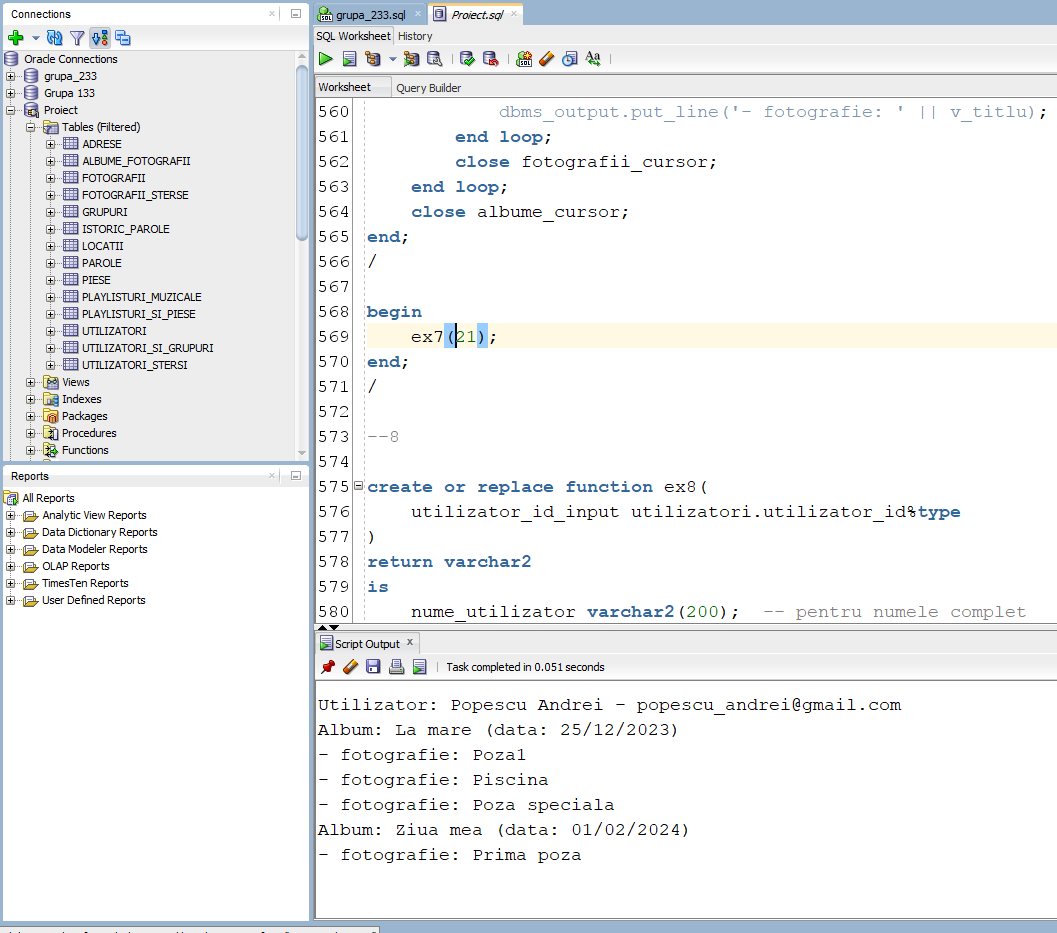
/

begin

ex7(21);

end;

/



# 8. Subprogram stocat independent de tip funcție care sa utilizeze 3 tabele într-o singură comandă

Cerință: Să se afișeze numele și locația utilizatorului cu id-ul dat ca parametru. (3 tabele intr-o comanda)

create or replace function ex8(utilizator\_id\_input utilizatori.utilizator\_id%type)

return varchar2

is

adresa\_completa varchar2(400);

begin

select u.nume || ' ' || u.prenume || ' - ' ||

a.strada || ', nr. ' || a.numar\_strada || ', ap. ' || a.apartament

into adresa\_completa

from utilizatori u

join locatii l on u.locatie\_id = l.locatie\_id

join adrese a on l.adresa\_id = a.adresa\_id

where u.utilizator\_id = utilizator\_id\_input;

return adresa\_completa;

exception

when no\_data\_found then

return 'Utilizatorul nu are o locatie asociata.';

when too\_many\_rows then

return 'Utilizatorul are mai multe locatii asociate.';

when others then

return 'Eroare necunoscuta';

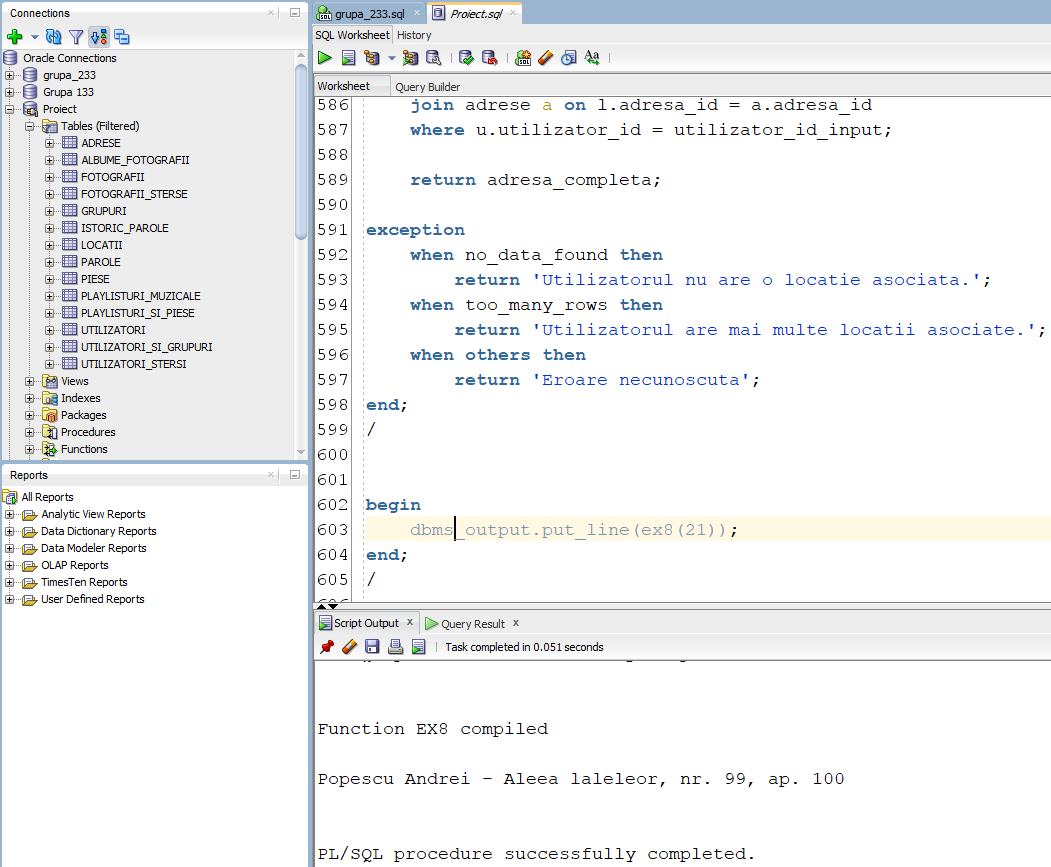
end;

/

begin

dbms\_output.put\_line(ex8(21));

end;



# 9. Subprogram stocat independent de tip procedură cu 2 parametrii care să utilizeze 5 tabele într-o comandă

Cerință: Să se afișeze utilizatorul curent și piesele salvate de acesta, la fiecare piesă afișându-se orașul si parola utilizatorului. (2 parametrii, 2 exceptii proprii, 5 tabele intr o comanda)

create or replace procedure ex9(

p\_utilizator\_id utilizatori.utilizator\_id%type,

p\_album\_nume albume\_fotografii.album\_nume%type

)

is

-- Declaratia exceptiilor proprii

ex\_utilizator\_inexistent exception;

ex\_album\_inexistent exception;

-- Variabile pentru stocarea informatiilor utilizatorului

v\_nume\_utilizator varchar2(200);

v\_email\_utilizator varchar2(200);

v\_album\_id albume\_fotografii.album\_id%type;

-- Cursor complex care integrează informații din 5 tabele

cursor detalii\_cursor is

select

u.nume || ' ' || u.prenume as utilizator\_nume,

p.piesa\_nume as piesa\_nume,

pa.parola\_nume as parola,

l.oras as locatie

from utilizatori u

join playlisturi\_si\_piese pp on u.utilizator\_id = pp.utilizator\_id

join piese p on pp.piesa\_id = p.piesa\_id

join parole pa on u.parola\_id = pa.parola\_id

join locatii l on u.locatie\_id = l.locatie\_id

where u.utilizator\_id = p\_utilizator\_id;

-- Variabile pentru cursor

v\_utilizator\_nume varchar2(200);

v\_piesa\_nume piese.piesa\_nume%type;

v\_fotografie\_titlu fotografii.titlu%type;

v\_album\_nume albume\_fotografii.album\_nume%type;

v\_parola parole.parola\_nume%type;

v\_oras locatii.oras%type;

begin

-- Verificam daca utilizatorul exista

begin

select nume || ' ' || prenume, email

into v\_nume\_utilizator, v\_email\_utilizator

from utilizatori

where utilizator\_id = p\_utilizator\_id;

exception

when no\_data\_found then

raise ex\_utilizator\_inexistent;

end;

-- Verificam daca albumul apartine utilizatorului

begin

select album\_id

into v\_album\_id

from albume\_fotografii

where utilizator\_id = p\_utilizator\_id

and album\_nume = p\_album\_nume;

exception

when no\_data\_found then

raise ex\_album\_inexistent;

end;

-- Afisam informatiile

dbms\_output.put\_line('Utilizator: ' || v\_nume\_utilizator || ' - ' || v\_email\_utilizator);

dbms\_output.put\_line('Detalii piese:');

open detalii\_cursor;

loop

fetch detalii\_cursor into v\_utilizator\_nume, v\_piesa\_nume, v\_parola, v\_oras;

exit when detalii\_cursor%notfound;

dbms\_output.put\_line('Piesa: ' || v\_piesa\_nume);

dbms\_output.put\_line('Oras: ' || v\_oras);

dbms\_output.put\_line('Parola: ' || v\_parola);

dbms\_output.put\_line('');

end loop;

close detalii\_cursor;

exception

when ex\_utilizator\_inexistent then

dbms\_output.put\_line('Eroare: Utilizatorul cu ID-ul ' || p\_utilizator\_id || ' nu exista.');

when ex\_album\_inexistent then

dbms\_output.put\_line('Eroare: Albumul "' || p\_album\_nume || '" nu apartine utilizatorului specificat.');

when others then

dbms\_output.put\_line('A aparut o eroare.');

end;

/

begin

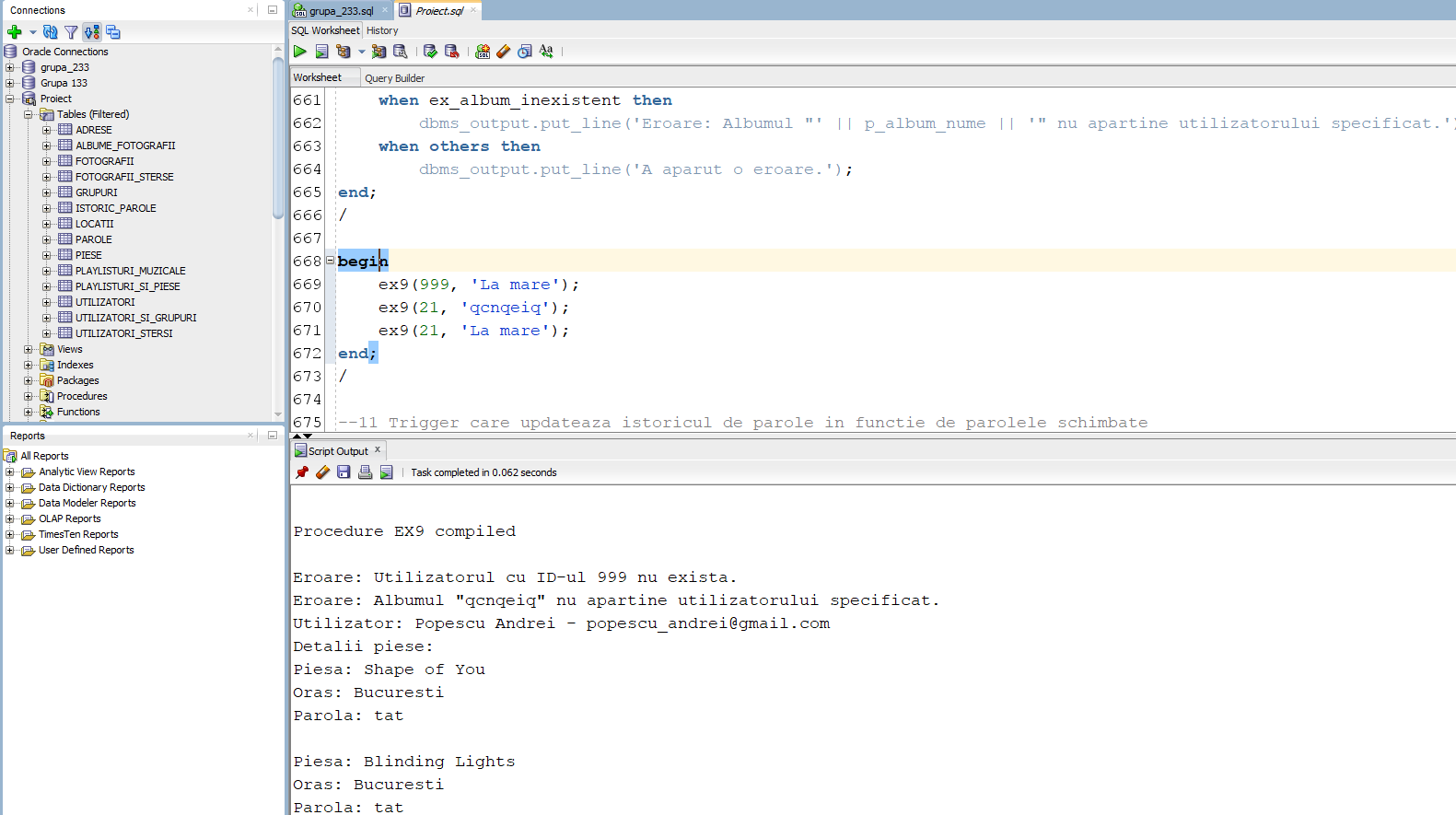
ex9(999, 'La mare');

ex9(21, 'qcnqeiq');

ex9(21, 'La mare');

end;

/



# 10. Trigger de tip LMD la nivel de comandă

Cerință: Să se creeze un trigger de tip LMD la nivel de comandă, care după inserarea in tabela fotografii, să insereze numarul total de fotografii și data din acel moment.

create sequence seq\_trigger

start with 1

increment by 1;

create table aux\_fotografii (

id number(5) primary key,

numar\_fotografii number(5) not null,

data\_operatie date default sysdate

);

create or replace trigger trg\_insert\_fotografii

after insert on fotografii

declare

total\_fotografii number;

begin

select count(\*)

into total\_fotografii

from fotografii;

insert into aux\_fotografii (id, numar\_fotografii, data\_operatie)

values (seq\_trigger.nextval, total\_fotografii, sysdate);

end;

/

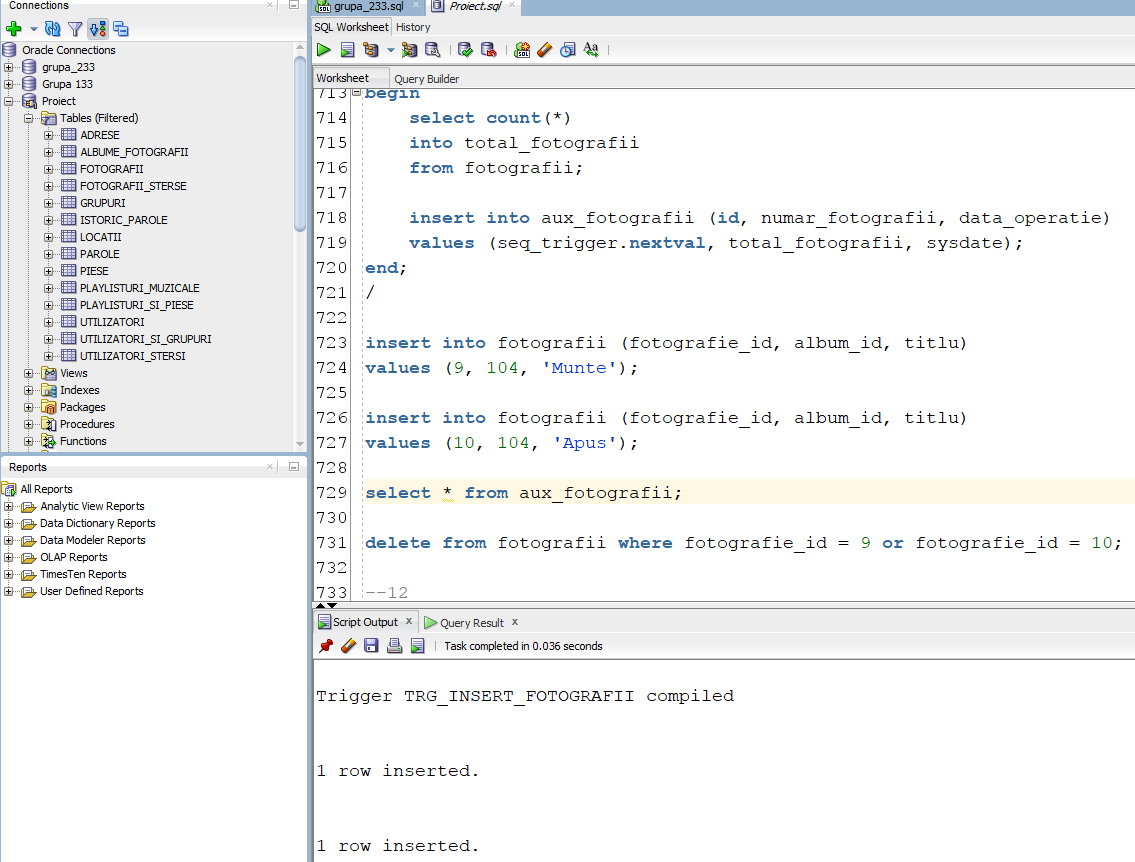
insert into fotografii (fotografie\_id, album\_id, titlu)

values (9, 104, 'Munte');

insert into fotografii (fotografie\_id, album\_id, titlu)

values (10, 104, 'Apus');

select \* from aux\_fotografii;



# 11. Trigger de tip LMD la nivel de linie

Cerință: Să se creeze un trigger de tip LMD la nivel de linie, care după schimbarea unei parole, insereză in tabela istoric\_parole datele corespunzătoare despre vechea parolă.

create or replace trigger trg\_schimbare\_parola

after update of parola\_id on utilizatori

for each row

begin

-- verificam daca parola s-a schimbat

if :old.parola\_id is not null and :new.parola\_id != :old.parola\_id then

-- adaugam in istoric noua parola

insert into istoric\_parole (utilizator\_id, data\_schimbare, parola\_id)

values (:new.utilizator\_id, sysdate, :new.parola\_id);

end if;

end;

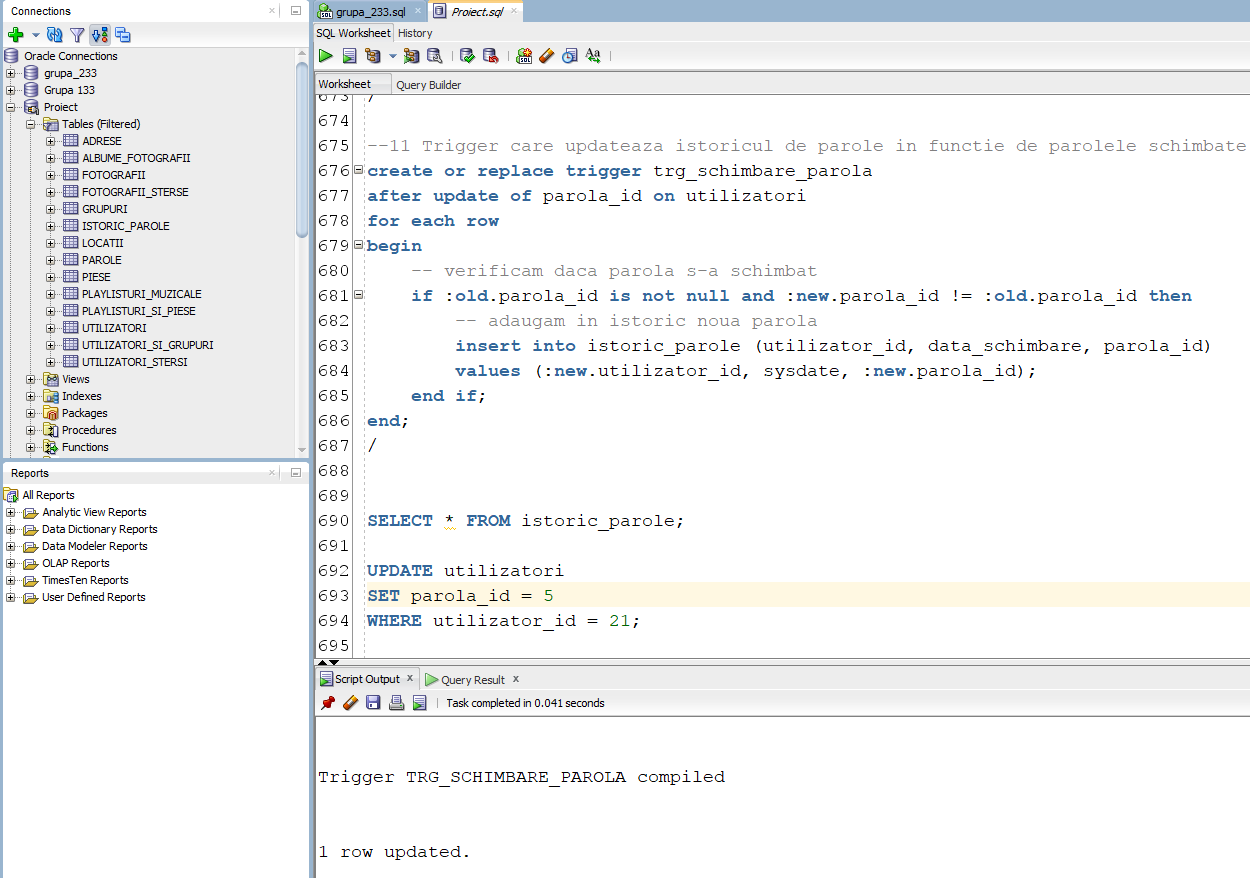
/

SELECT \* FROM istoric\_parole;

UPDATE utilizatori

SET parola\_id = 5

WHERE utilizator\_id = 21;



# 12. Trigger de tip LDD

Cerință: Să se creeze un trigger de tip LLD care după modificarea unui tabele, datele despre modificare sunt salvate într-o tabelă auxiliară.

create sequence seq\_aux\_piese

start with 1

increment by 1;

create table aux\_piese(

id number(5) primary key,

detalii varchar(200)

);

create or replace trigger trg\_alter\_aux\_piese

after alter on schema

begin

insert into aux\_piese(id, detalii)

values (seq\_aux\_piese.nextval, 'Tabela PIESE a fost modificata');

end;

/

select \* from piese;

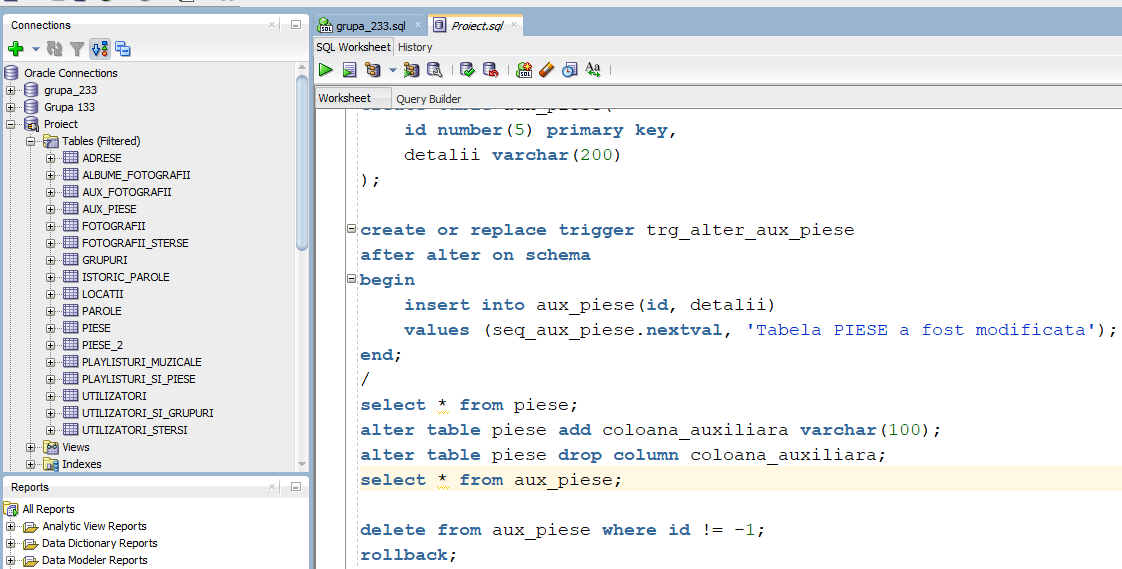
alter table piese add coloana\_auxiliara varchar(100);

alter table piese drop column coloana\_auxiliara;

select \* from aux\_piese;

delete from aux\_piese where id != -1;

rollback;



# 13. Pachet care să includă 2 tipuri de date complexe, 2 funcții și 2 proceduri

create sequence seq\_playlist

start with 1

increment by 1

nocache;

create or replace package ex13\_pachet as

-- Pentru ID-urile pieselor

type t\_piese\_imbricat is table of number;

-- Pentru piesele unui playlist

type t\_piese\_indexat is table of number index by pls\_integer;

-- Functie pentru obtinerea playlisturilor unui utilizator (pe baza ID-urilor pieselor)

function obtine\_playlisturi(utilizator\_id\_input number) return t\_piese\_imbricat;

-- Functie pentru numararea pieselor dintr-un playlist

function numara\_piese(playlist\_id\_input number) return number;

-- Procedura pentru adaugarea unui playlist nou (folosind ID-uri de piese)

procedure adauga\_playlist(utilizator\_id\_input number, playlist\_nume\_input varchar2, lista\_piese t\_piese\_imbricat);

-- Procedura pentru actualizarea pieselor dintr-un playlist (folosind ID-uri de piese)

procedure actualizeaza\_piese(playlist\_id\_input number, lista\_piese t\_piese\_indexat);

end ex13\_pachet;

/

create or replace package body ex13\_pachet as

function obtine\_playlisturi(utilizator\_id\_input number) return t\_piese\_imbricat is

playlisturi t\_piese\_imbricat;

begin

select ps.piesa\_id

bulk collect into playlisturi

from playlisturi\_si\_piese ps

where ps.utilizator\_id = utilizator\_id\_input;

return playlisturi;

exception

when no\_data\_found then

return t\_piese\_imbricat();

when others then

return t\_piese\_imbricat();

end;

-- functia pentru numararea pieselor dintr-un playlist

function numara\_piese(playlist\_id\_input number) return number is

total\_piese number;

begin

select count(\*)

into total\_piese

from playlisturi\_si\_piese

where playlist\_id = playlist\_id\_input;

return total\_piese;

exception

when no\_data\_found then

return 0;

when others then

return -1;

end;

-- procedura pentru adaugarea unui playlist nou (folosind ID-uri de piese)

procedure adauga\_playlist(utilizator\_id\_input number, playlist\_nume\_input varchar2, lista\_piese t\_piese\_imbricat) is

new\_playlist\_id number;

begin

-- adaugam playlistul in tabela playlisturi\_muzicale

select seq\_playlist.nextval into new\_playlist\_id from dual;

insert into playlisturi\_muzicale (playlist\_id, playlist\_nume)

values (new\_playlist\_id, playlist\_nume\_input);

-- adaugam piesele asociate in tabela playlisturi\_si\_piese

for i in 1 .. lista\_piese.count loop

insert into playlisturi\_si\_piese (piesa\_id, playlist\_id, utilizator\_id)

values (lista\_piese(i), new\_playlist\_id, utilizator\_id\_input);

end loop;

dbms\_output.put\_line('Playlistul a fost adaugat cu succes.');

exception

when others then

dbms\_output.put\_line('Eroare la adaugarea playlistului' || sqlerrm);

end;

-- procedura pentru actualizarea pieselor dintr-un playlist

procedure actualizeaza\_piese(playlist\_id\_input number, lista\_piese t\_piese\_indexat) is

utilizator\_id\_cautat number;

begin

-- Obtinem id-ul utilizatorului care detine playlist-ul

select utilizator\_id

into utilizator\_id\_cautat

from playlisturi\_si\_piese

where playlist\_id = playlist\_id\_input

and rownum = 1;

-- Stergem piesele existente din playlist

delete from playlisturi\_si\_piese

where playlist\_id = playlist\_id\_input;

-- Adaugam piesele noi

for i in 1 .. lista\_piese.count loop

insert into playlisturi\_si\_piese (piesa\_id, playlist\_id, utilizator\_id)

values (lista\_piese(i), playlist\_id\_input, utilizator\_id\_cautat);

end loop;

dbms\_output.put\_line('Playlistul a fost actualizat cu succes.');

exception

when no\_data\_found then

dbms\_output.put\_line('Playlistul nu are niciun utilizator asociat.');

when others then

dbms\_output.put\_line('Eroare la actualizarea pieselor: ' || sqlerrm);

end;

end ex13\_pachet;

/

declare

-- Crearea unui tablou imbricat pentru piesele playlistului nou

piese\_playlist\_nou ex13\_pachet.t\_piese\_imbricat := ex13\_pachet.t\_piese\_imbricat(1, 2, 3, 4);

-- Crearea unui tablou indexat pentru piesele playlistului actualizat

piese\_actualizate ex13\_pachet.t\_piese\_indexat;

begin

ex13\_pachet.adauga\_playlist(21, 'playlist nou', piese\_playlist\_nou);

piese\_actualizate(1) := 5;

piese\_actualizate(2) := 6;

piese\_actualizate(3) := 7;

ex13\_pachet.actualizeaza\_piese(4, piese\_actualizate);

end;

/

select \* from playlisturi\_si\_piese;

select \* from playlisturi\_muzicale;

select \* from utilizatori;

delete from playlisturi\_muzicale where playlist\_nume like 'playlist nou';

