Tema 1 ex 1: begin

for t in(select table\_name from user\_tables) loop

for c in (select constraint\_name from user\_constraints where constraint\_type='R' AND table\_name=t.table\_name) loop

execute immediate 'ALTER TABLE ' || t.table\_name || ' DROP CONSTRAINT ' || c.constraint\_name;

end loop;

execute immediate 'DROP TABLE ' || t.table\_name;

end loop;

end;

ex 2: begin

for i in 1..5 loop

insert into departments (department\_id, department\_name, manager\_id, location\_id)

values(i, 'Departamentul ' || i, i-1, i\*25);

end loop;

end;

tema 2: with nrcantece as (

select a.gen, count(ist.melodie\_id) as nr\_cantece

from istoric\_melodii ist

join melodie m on ist.melodie\_id = m.melodie\_id

join album a on m.album\_id=a.album\_id

group by a.gen

)

select gen, nr\_cantece

from nrcantece

where nr\_cantece = (

select max(nr\_cantece)

from nrcantece);

Tema 2:

Ex1: UTILIZATOR\_URMARESTE\_ARTIST: relatie de tip many-to-many care leaga entitatile UTILIZATOR si ARTIST si care exprima legatura dintre cele doua astfel: daca un utilizator urmareste un artist, va putea gasi mai usor lansarile (albumele si melodiile) acestuia. Relatia are cardinalitate minima 0:0 si cardinalitate maxima m:n.

UTILIZATOR: persoana care utilizeaza aceasta aplicatie prin intermediul contului pe care il creeaza. Cheia primara este utilizator\_id.

ARTIST: persoana care are posibilitatea de a incarca melodii si albume in aplicatie pentru a fi ascultate de utilizatori. Aceasta entitate are drept cheie primara artist\_id.

Ex3:

Entitatea UTILIZATOR are ca atribute:

→ utilizator\_id (**cheie primară**) = variabilă de tip întreg, care reprezintă id-ul unui utilizator.

→ nume\_utilizator = variabila de tip caracter, de lungime maximă 50, nenulă, unică, care reprezintă numele pe care ceilalți utilizatori îl văd în aplicație.

→ parola = variabilă de tip caracter, de lungime maximă 128, nenulă, care reprezintă parola cu care utilizatorul se autentifică în cont.

→ email = variabilă de tip caracter, de lungime maximă 150, care reprezintă adresa de e-mail a utilizatorului.

→data\_nastere = variabilă de tip dată calendaristică, nenulă, care reprezintă data nașterii utilizatorului.

→ tara = variabilă de tip caracter, de lungime maximă 2, nenulă, care reprezintă codul țării unui utilizator, conform Anexei A6 europene, “Codurile țărilor și teritoriilor”

Entitatea ARTIST are ca atribute:

→ artist\_id (**cheie primară**) = variabilă de tip întreg, care reprezintă id-ul unui artist.

→ nume = variabilă de tip caracter, nenulă, de lungime maximă 100, care reprezintă numele artistului/trupei.

→ varsta = variabilă de tip întreg, care poate fi NULL pentru cazul trupelor, și reprezintă vârsta artistului, acolo unde e cazul.

→ descriere = variabilă de tip caracter, de dimensiune maximă 255, reprezintă o mică descriere pe care artistul/trupa o poate avea.

Relația UTILIZATOR\_urmărește\_ARTIST are ca atribute: utilizator\_id = variabilă de tip întreg, care reprezintă id-ul unui utilizator, atributul trebuie să corespundă unei chei primare din tabelul UTILIZATOR. artist\_id = variabilă de tip întreg, care reprezintă id-ul unui artist. Atributul trebuie să corespundă unei chei primare din tabelul ARTIST. Acest tabel present în diagrama conceptuală are drept **cheie primară**, cheia compusă din cele două attribute: utilizator\_id și artist\_id, unde acestea sunt și chei externe.

e) creare tabele + chei + constrangeri

--tabel UTILIZATOR

CREATE TABLE UTILIZATOR(

utilizator\_id INT PRIMARY KEY,

nume\_utilizator VARCHAR(50) UNIQUE,

parola VARCHAR(128) NOT NULL,

email VARCHAR(150),

data\_nastere DATE NOT NULL,

tara VARCHAR(2) NOT NULL

);

--table ARTIST

CREATE TABLE ARTIST(

artist\_id INT PRIMARY KEY,

nume VARCHAR(100),

varsta INT,

descriere VARCHAR(255)

);

--tabel ALBUM

CREATE TABLE ALBUM (

album\_id INT PRIMARY KEY,

gen VARCHAR(50) NOT NULL,

pret DECIMAL(10,2) CHECK (pret>0),

titlu VARCHAR(100) NOT NULL,

data\_lansare DATE NOT NULL,

artist\_id INT NOT NULL,

CONSTRAINT album\_artist\_fk FOREIGN KEY (artist\_id) REFERENCES ARTIST

(artist\_id)

--dam nume pt fk ca sa putem identifica erori

);

--tabel UTILIZATOR\_apreciaza\_ALBUM

CREATE TABLE UTILIZATOR\_APRECIAZA\_ALBUM(

utilizator\_id INT,

album\_id INT,

PRIMARY KEY(utilizator\_id, album\_id),

FOREIGN KEY(utilizator\_id) REFERENCES UTILIZATOR(utilizator\_id),

FOREIGN KEY(album\_id) REFERENCES ALBUM (album\_id)

);

--tabel ARTIST\_LANSEAZA\_ALBUM

CREATE TABLE ARTIST\_LANSEAZA\_ALBUM(

artist\_id INT,

album\_id INT,

PRIMARY KEY(artist\_id, album\_id),

FOREIGN KEY(artist\_id) REFERENCES ARTIST (artist\_id),

FOREIGN KEY(album\_id) REFERENCES ALBUM(album\_id)

);

--tabel UTILIZATOR\_URMARESTE\_ARTIST

CREATE TABLE UTILIZATOR\_URMARESTE\_ARTIST(

utilizator\_id INT,

artist\_id INT,

PRIMARY KEY (utilizator\_id, artist\_id),

FOREIGN KEY(utilizator\_id) REFERENCES UTILIZATOR(utilizator\_id),

FOREIGN KEY(artist\_id) REFERENCES ARTIST(artist\_id)

);

--tabel MELODIE

CREATE TABLE MELODIE(

melodie\_id INT PRIMARY KEY,

titlu VARCHAR(255) NOT NULL,

descriere VARCHAR(255),

durata NUMBER(3,0) NOT NULL,

data\_lansare DATE NOT NULL,

album\_id INT NOT NULL,

artist\_id INT,

CONSTRAINT song\_artist\_fk FOREIGN KEY (artist\_id) REFERENCES

ARTIST(artist\_id),

CONSTRAINT song\_album\_fk FOREIGN KEY (album\_id) REFERENCES

ALBUM(album\_id)

);

--tabel ARTIST\_CANTA\_MELODIE

CREATE TABLE ARTIST\_CANTA\_MELODIE (

artist\_id INT,

melodie\_id INT,

PRIMARY KEY(artist\_id, melodie\_id),FOREIGN KEY (artist\_id) REFERENCES ARTIST (artist\_id),

FOREIGN KEY (melodie\_id) REFERENCES MELODIE(melodie\_id)

);

--tabel PLAYLIST

CREATE TABLE PLAYLIST (

playlist\_id INT PRIMARY KEY,

titlu VARCHAR(255) NOT NULL,

data\_creare DATE NOT NULL

);

--TABEL MELODIE\_IN\_PLAYLIST\_UTILIZATOR

CREATE TABLE MELODIE\_IN\_PLAYLIST\_UTILIZATOR(

melodie\_id INT,

playlist\_id INT,

utilizator\_id INT,

data\_adaugare DATE NOT NULL, --data adaugarii melodiei in playlist-ul userului

PRIMARY KEY (melodie\_id, playlist\_id, utilizator\_id),

FOREIGN KEY(melodie\_id) REFERENCES MELODIE(melodie\_id),

FOREIGN KEY (playlist\_id) REFERENCES PLAYLIST(playlist\_id),

FOREIGN KEY (utilizator\_id) REFERENCES UTILIZATOR(utilizator\_id)

);

--tabel UTILIZATOR\_APRECIAZA\_PLAYLIST

CREATE TABLE UTILIZATOR\_APRECIAZA\_PLAYLIST(

utilizator\_id INT,

playlist\_id INT,

PRIMARY KEY (utilizator\_id, playlist\_id),

FOREIGN KEY (utilizator\_id) REFERENCES UTILIZATOR(utilizator\_id),FOREIGN KEY(playlist\_id) REFERENCES PLAYLIST(playlist\_id)

);

--tabel ISTORIC\_MELODII

CREATE TABLE ISTORIC\_MELODII(

istoric\_melodii\_id INT PRIMARY KEY,

utilizator\_id INT,

melodie\_id INT,

data\_ascultare DATE NOT NULL,

FOREIGN KEY (utilizator\_id) REFERENCES UTILIZATOR(utilizator\_id),

FOREIGN KEY(melodie\_id) REFERENCES MELODIE(melodie\_id)

);

--table PODCAST

CREATE TABLE PODCAST (

podcast\_id INT PRIMARY KEY,

nume VARCHAR(255) NOT NULL,

nr\_episoade INT CHECK (nr\_episoade >0),

nume\_autor VARCHAR(100) NOT NULL

);

--tabel EPISOD

CREATE TABLE EPISOD(

episod\_id INT PRIMARY KEY,

podcast\_id INT,

titlu VARCHAR(120) NOT NULL,

FOREIGN KEY (podcast\_id) REFERENCES PODCAST(podcast\_id)

);

--tabel PODCAST\_ISTORIC

CREATE TABLE PODCAST\_ISTORIC(

podcast\_istoric\_id INT PRIMARY KEY,

utilizator\_id INT,

episod\_id INT,

data\_ascultare DATE NOT NULL,

FOREIGN KEY (utilizator\_id) REFERENCES UTILIZATOR(utilizator\_id),

FOREIGN KEY (episod\_id) REFERENCES EPISOD(episod\_id)

);