

**EGE UNIVERSITY**

**ENGINEERING FACULTY**

**COMPUTER ENGINEERING**

**DATABASE MANAGEMENT**

**2021–2022 FALL SEMESTER**

**MIDTERM PROJECT REPORT**

**HAZIRLAYANLAR**

05190000072, Tuğcan Topaloğlu

05180000047, Elifnaz Acun

05180000113, Simge Merve Yaşbay

05180000081, Özlem Çelebi

İçindekiler

[1. Analysis 2](#_Toc94715499)

[1.1 Brief Explanation 2](#_Toc94715500)

[2.1 Analysis Report for Each Web application 2](#_Toc94715501)

[2.1.a What is the aim of each application? 2](#_Toc94715502)

[2.1.b What are the main entities of them? 2](#_Toc94715503)

[2.1.c What are the characteristics of each entity? 2](#_Toc94715504)

[2.1.d What relationships exists among the entities? 3](#_Toc94715505)

[2.1.e What are the constraints related to entities, their characteristics and the relationships ssssamong them? 4](#_Toc94715506)

[2. Design - Conceptual Design 5](#_Toc94715507)

[3. Design - Logical Model 6](#_Toc94715508)

[3.1 Mapping 6](#_Toc94715509)

[3.2 Relational Model 9](#_Toc94715510)

[4. Implementation - Physical Model 10](#_Toc94715511)

[4.1 Tables 10](#_Toc94715512)

[4.2 Tuples 20](#_Toc94715513)

[4.3 Triggers 21](#_Toc94715514)

[4.4 Check Constraints 23](#_Toc94715515)

[4.5 SQL Statements 26](#_Toc94715516)

[4.5.a INSERT/DELETE/UPDATE Statements 26](#_Toc94715517)

[4.5.b Select Statements 27](#_Toc94715518)

[4.5.c Original Select Statements 31](#_Toc94715519)

# 1. Analysis

## 1.1 Brief Explanation

If we look at these two programs independently for the first time, LinkedIn is a business networking application that connects registered individuals, corporations, and schools all around the world. Job advertising, employment applications, job contacts, and talents, work experiences, interests, and other fields relevant to each user are all included in the application, as are profiles in the form of a CV.

Moodle, on the other hand, caters to a more specialized audience and allows teachers to require students to register for the course using the passwords they have created, and it incorporates quizzes, exams, homework, and other course-related activities.

## 2.1 Analysis Report for Each Web application

### 2.1.a What is the aim of each application?

LinkedIn's goal is to link professionals all over the world in order to create a more efficient and productive business network by allowing them to interact. Moodle's goal is to provide a unique educational environment for educators and students through its use and control, resulting in a more efficient teaching platform.

### 2.1.b What are the main entities of them?

Common main entities of both Db\_User , User\_Profile , College , Faculty , Department , Transcript.

In addition to these, the main entities of LinkedIn Company , Company\_Profile , Collection , Job\_Offer , Post , Achievement.

In addition to these, the main entities of Moodle Course , Grading\_Req.

### 2.1.c What are the characteristics of each entity?

Db\_User entity has unique user\_id,username and password. The user can be of Student,Teacher and Worker. It has flag attributes that control each of them. If the user is a worker it has sector,career title and work time. If the user is a teacher it has teacher degree.

User\_Profile entity has unique user\_profile\_id , full name , unique mail , address , unique phone , sex and birth date.

Company entity entity has unique company\_id and mgr\_id.

Company\_Profile entity has unique company\_profile\_id,unique company name,company location, and unique company phone.

Transcript entity has unique transcript no , type , taken date and gpa.

College entity has unique college id , college name , location and phone.

Faculty entity has unique faculty id , name , location and phone.

Department entity has unique id,phone , name and location.

Course entity has unique id,name and description.

Grading\_Req entity has unique id and type.One Grading\_req can be only one of four types.These are Quiz,Project,Exam and Homework.Each of them has grade attribute.

Achievement entity has unique id,type and date. One Achievement can be only one of four types. These are Test Score , Willing Project , Language and Certificate. Each of them has name attribute and Test Score has score attribute.

Collection entity has unique id. Collection includes two type data.These are Job Offer and Post. Job Offer has unique id , title and location. Post has unique id.

### 2.1.d What relationships exists among the entities?

Each Db\_User must has a User\_Profile.Each User\_Profile must belong a Db\_User.

Each User\_Profile can display more than one profile, and each User\_Profile can be viewed by more than one User\_Profile.Each User\_Profile can connect with multiple User\_Profile.

Each Db\_User can save more than one Collection. Each Collection must necessarily be registered by a Db\_User.

Each Db\_User can apply to or view more than one Job\_Offer. Each Job\_Offer can be applied or viewed by more than one Db\_User.

Each Db\_User can share, like, comment on more than one Post. Each Post can be liked, shared, commented on by more than one Db\_User.

Each Db\_User can achieve more than one Achievement. An Achievement can belong to more than one Db\_User.

Each Db\_User can upload more than one Private\_File. One Private\_File must necessarily be uploaded by a Db\_User.

Each Db\_User can has more than one Transcript. One Transcript can only belong to one Db\_User.

One Transcript can taken by only one College. Each College can take more than one Transcript.

Each Student can study only one College. Each College can has more than one Student.

Each Teacher can work only one College. Each College can has more than one Teacher.

Each College can have more than one Faculty. Each Faculty can belong to only one College.

Each Faculty can has more than one Department. Each Department must necessarily and only belong to one Faculty.

Each Department can give one or more Course. Each Course is provided by only one Department.

Each Student can enroll in more than one Course. Each Course can receive more than one Student registration.

Each Teacher can teach more than one Course. Each Course must be taught by one or more Teacher.

Each Course must has one or more Grading\_Req. Each Grading\_Req must belong to only one Course.

Every Worker can has a career in a Company. Each Company can has more than one Worker.

Each Company must has a Company\_Profile. Each Company\_Profile must belong to a Company.

Each Company can offers more than one Job\_Offer. Each Job\_Offer must be offered by only one Company.

### 2.1.e What are the constraints related to entities, their characteristics and the relationships among them?

Db\_User’s user\_id attiribute must be initialized between 100000 and 199999.

Transcript’s transcprit\_no attribute must be initialized between 200000 and 299999.

College’s college\_id attribute must be initialized between 300000 and 399999.

Faculty’s faculty\_id attribute must be initialized between 400000 and 499999.

Department’s department\_id attribute must be initialized between 500000 and 599999.

Course’s course\_id attribute must be initialized between 600000 and 699999.

Grading\_Req’s grading\_req\_id attribute must be initialized between 700000 and 799999.

Company’s company\_id attribute must be initialized between 800000 and 899999.

User\_Profile’s user\_profile\_id attribute must be initialized between 900000 and 999999.

Company\_Profile’s company\_profile\_id attribute must be initialized between 700000 and 799999.

Achievement’s achievement\_id attribute must be initialized between 110000 and 119999.

Collection’s cllection\_id attribute must be initialized between 120000 and 129999.

Job\_Offer’s job\_offer\_id attribute must be initialized between 130000 and 139999.

Post’s post\_id attribute must be initialized between 140000 and 149999.

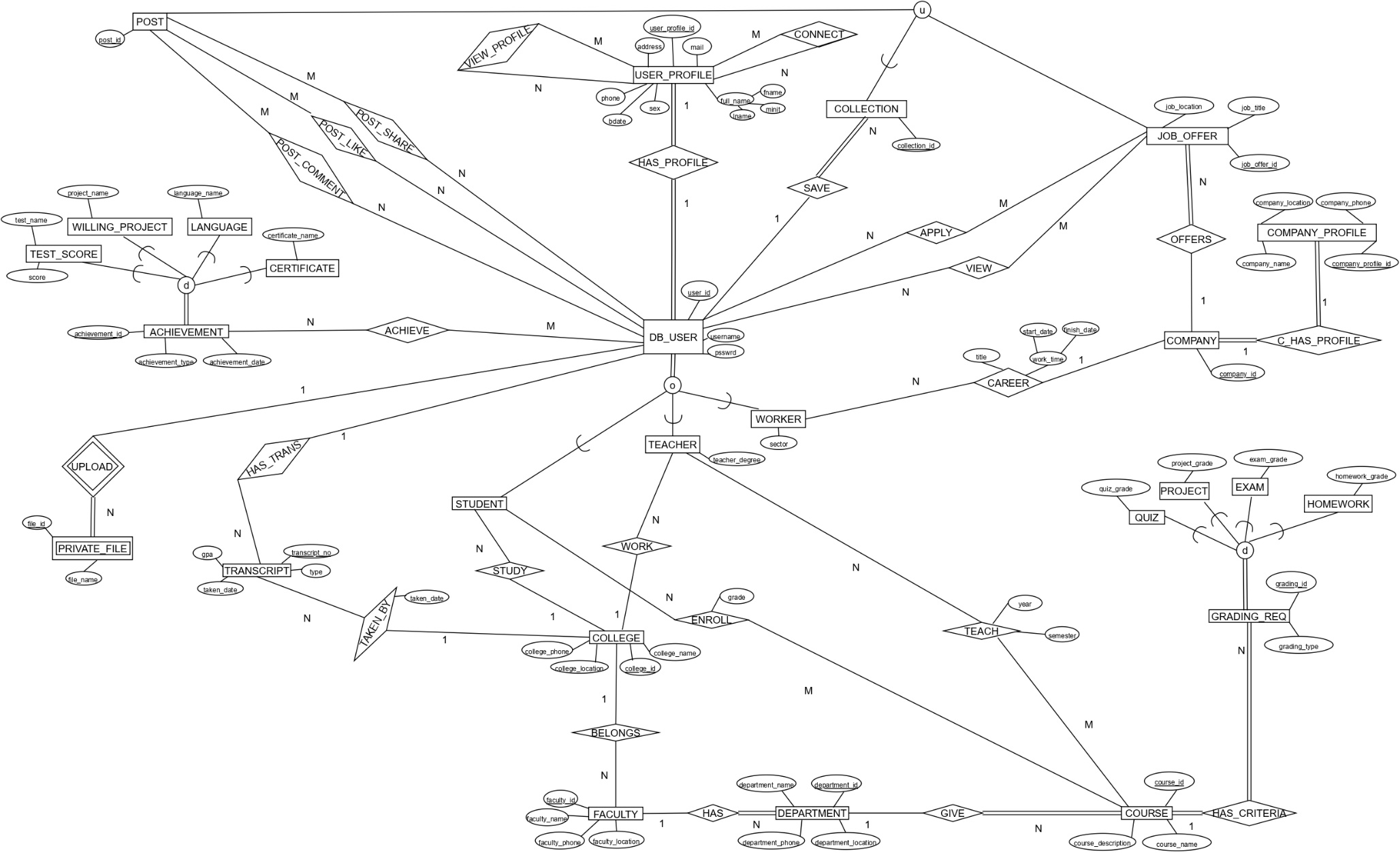
Private\_File’s file\_id attribute must be initialized between 150000 and 159999.

Grading types can be Exa, Project, Homework or Quiz.

Achievements types can be Willing Project, Language, Test Score or Certificate.

Collection can be both job offer or post.

# 2. Design - Conceptual Design



# 3. Design - Logical Model

## 3.1 Mapping

1.ITERATION

1-)

TRANSCRIPT(transcript\_no,type,taken\_date,gpa)

COLLEGE(college\_id,college\_name,college\_location,college\_phone)

FACULTY(faculty\_id,faculty\_name,faculty\_location,faculty\_phone)

DEPARTMENT(department\_id,department\_phone,department\_name,department\_location)

COURSE(course\_id,course\_name,course\_description)

COMPANY(company\_id,mgr\_id)

COMPANY\_PROFILE(c\_profile\_id,company\_name,comapny\_loc,company\_phone)

USER\_PROFILE(user\_profile\_id,fname,minit,lname,mail,address,phone,sex,bdate)

2-)

3-) 1 to 1

COMPANY(company\_id,mgr\_id,c\_profile\_id) C\_HAS\_PROFILE

4-) 1 TO N

FACULTY(faculty\_id,faculty\_name,faculty\_location,faculty\_phone,faculty\_college\_id) BELONGS

DEPARTMENT(department\_id,department\_phone,department\_name,department\_location,department\_faculty\_id) HAS

COURSE(course\_id,course\_name,course\_description,course\_department\_id) GIVE

TRANSCRIPT(transcript\_no,type,taken\_date,gpa,t\_college\_id) TAKEN\_BY

5-) M to N

VIEW\_PROFILE(user\_profile\_id,user\_viewer\_id)

CONNECT(user\_profile\_id,user\_connect\_id)

6-)

7-)

8-)

8.D.)

DB\_USER(user\_id,username,psswrd,student\_flag,teacher\_flag,worker\_flag,teacher\_degree,sector)

8.A.)

ACHIEVEMENT(achievemet\_id,date,achievement\_type)

TEST\_SCORE(achievemet\_id,score,test\_name)

WILLING\_PROJECT(achievemet\_id,project\_name)

LANGUAGE(achievement\_id,language\_name)

CERTIFICATE(achievement\_id,certificate\_name)

8.A.)

GRADING\_REQ(grading\_req\_id,grading\_type)

QUIZ(grading\_req\_id,quiz\_grade)

PROJECT(grading\_req\_id,project\_grade)

EXAM(grading\_req\_id,exam\_grade)

HOMEWORK(grading\_req\_id,homework\_grade)

9-)

COLLECTION(collection\_id)

POST(post\_id,post\_collection\_id)

JOB\_OFFER(job\_offer\_id,job\_collection\_id,job\_title,job\_location)

2.ITERATION

1-)

2-)

PRIVATE\_FILE(file\_id,private\_user\_id,file\_name) UPLOAD

3-) 1 to 1

DB\_USER(user\_id,username,psswrd,student\_flag,teacher\_flag,worker\_flag,teacher\_degree,sector,user\_profile\_id) HAS\_PROFILE

4-) 1 to N

TRANSCRIPT(transcript\_no,type,taken\_date,gpa,t\_college\_id,t\_user\_id) HAS\_TRANS

DB\_USER(user\_id,username,psswrd,student\_flag,teacher\_flag,worker\_flag,teacher\_degree,sector,user\_profile\_id,student\_college\_id) STUDY

DB\_USER(user\_id,username,psswrd,student\_flag,teacher\_flag,worker\_flag,teacher\_degree,sector,user\_profile\_id,student\_college\_id,teacher\_college\_id) WORK

DB\_USER(user\_id,username,psswrd,student\_flag,teacher\_flag,worker\_flag,teacher\_degree,sector,user\_profile\_id,student\_college\_id,teacher\_college\_id,worker\_company\_id,career\_title,start\_date,finish\_date) CAREER

JOB\_OFFER(job\_offer\_id,job\_collection\_id,job\_title,job\_location,job\_offer\_company\_id) OFFERS

COLLECTION(collection\_id,collection\_user\_id) SAVE

GRADING\_REQ(grading\_req\_id,grading\_type,grading\_req\_course\_id) HAS\_CRITERIA

5-) N TO M

LIKE(post\_id,user\_id)

COMMENT(post\_id,user\_id)

SHARE(post\_id,user\_id)

ENROLL(user\_id,course\_id,grade)

TEACH(user\_id,course\_id,year,semester)

ACHIEVE(achievement\_id,user\_id)

APPLY(job\_offer\_id,user\_id)

VIEW(job\_offer\_id,user\_id)

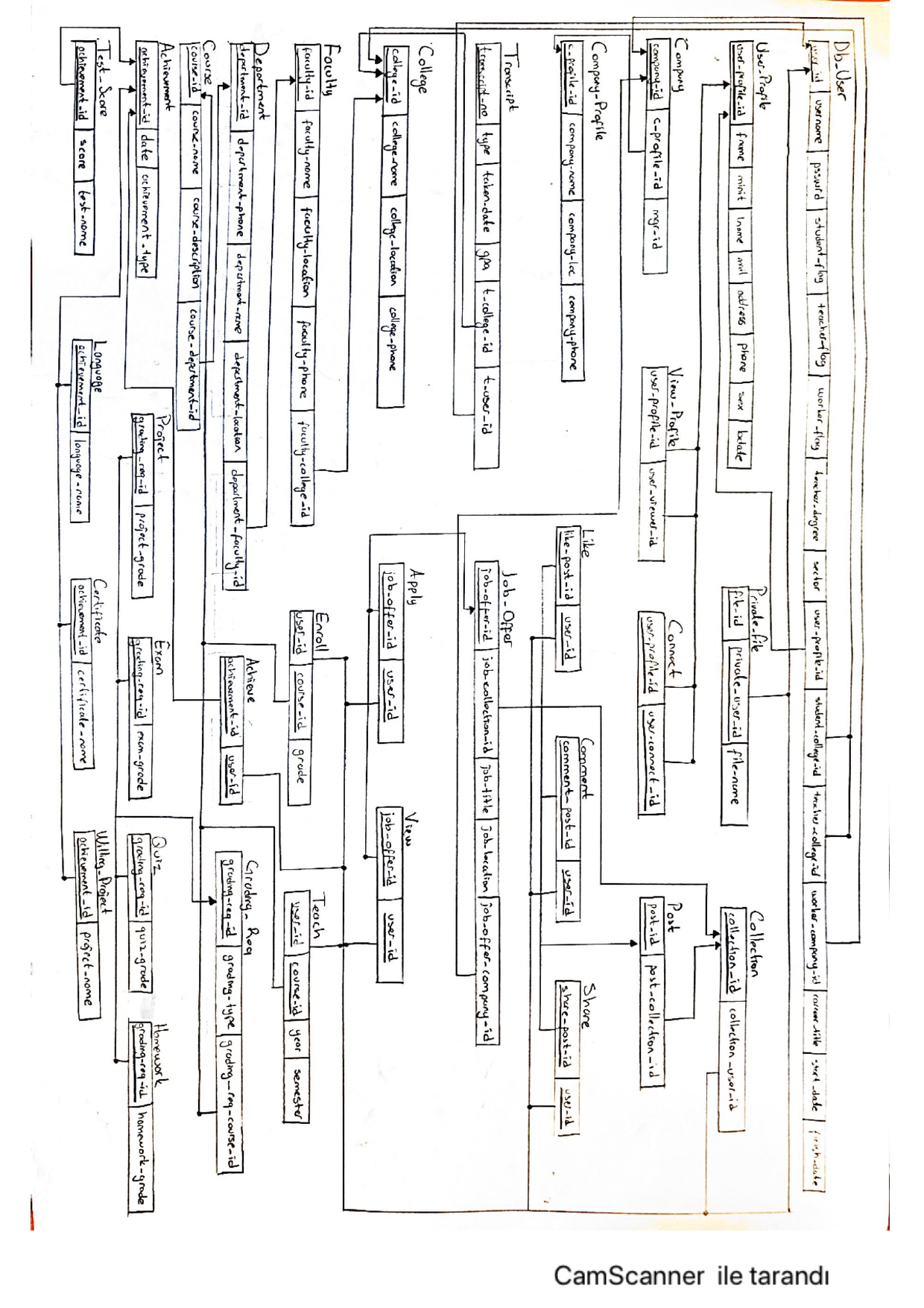
6-)

7-)

8-)

9-)

## 3.2 Relational Model



# 4. Implementation - Physical Model

Physical model implemented with POSTGRE SQL and pgAdmin 4.

All SQL codes can be found in Project folder. Tables can be found in “Tables.txt”. In order to maintain readibility all tuples are not included in tuples section and tuples can be found in “Tuples.txt”. Triggers can be found in “Triggers.txt”. Because of the disability of POSTGRE SQL instead of assertions we used check constraints and check constraints can be found in “CheckConstraints.txt”.Delete and Update insertions can be found in “DeleteUpdate.txt”. SQL statements can be found in “SelectStatements.txt”.

## 4.1 Tables

CREATE TYPE name AS(

Fname text,

Minit text,

Lname text

);

CREATE TYPE composite\_time AS(

start\_date Date,

finish\_date Date

);

CREATE TABLE College(

college\_id serial Primary Key NOT NULL,

college\_name text unique NOT NULL,

college\_location text,

college\_phone integer unique

);

CREATE TABLE Company\_Profile(

company\_profile\_id serial Primary Key NOT NULL,

company\_name text UNIQUE NOT NULL,

company\_location text,

company\_phone integer UNIQUE

);

CREATE TABLE User\_Profile(

user\_profile\_id serial Primary Key NOT NULL,

full\_name name,

mail text unique NOT NULL,

address text,

phone integer unique,

sex character,

BDate Date

);

CREATE TABLE Company(

company\_id serial Primary Key NOT NULL,

company\_profile\_id integer unique NOT NULL,

mgr\_id integer unique NOT NULL,

Foreign Key(company\_profile\_id) references Company\_Profile(company\_profile\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Db\_User(

user\_id serial Primary Key NOT NULL,

username text unique NOT NULL,

psswrd integer NOT NULL,

student\_flag boolean,

teacher\_flag boolean,

worker\_flag boolean,

sector text,

teacher\_degree text,

student\_college\_id integer,

teacher\_college\_id integer ,

worker\_company\_id integer ,

user\_profile\_id integer unique NOT NULL,

career\_title text,

work\_time composite\_time,

Foreign Key(student\_college\_id) references College (college\_id) ON DELETE CASCADE ON UPDATE CASCADE,

Foreign Key(teacher\_college\_id) references College (college\_id) ON DELETE CASCADE ON UPDATE CASCADE,

Foreign Key(worker\_company\_id) references Company (company\_id) ON DELETE CASCADE ON UPDATE CASCADE,

Foreign Key(user\_profile\_id) references User\_Profile (user\_profile\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Transcript(

transcript\_no serial Primary Key NOT NULL,

Type text NOT NULL,

taken\_date Date,

gpa numeric NOT NULL,

t\_user\_id integer NOT NULL,

t\_college\_id integer NOT NULL,

Foreign Key(t\_user\_id) references Db\_User(user\_id) ON DELETE CASCADE ON UPDATE CASCADE,

Foreign Key(t\_college\_id) references College(college\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Faculty (

faculty\_id serial Primary Key NOT NULL,

faculty\_name text NOT NULL,

faculty\_location text,

faculty\_phone integer unique,

faculty\_college\_id integer NOT NULL,

Foreign Key(faculty\_college\_id) references College(college\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Department (

department\_id serial Primary Key NOT NULL,

department\_faculty\_id integer NOT NULL,

department\_phone integer unique,

department\_name text NOT NULL,

department\_location text,

Foreign Key(department\_faculty\_id) references Faculty(faculty\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Course(

course\_id serial Primary Key NOT NULL,

course\_department\_id integer NOT NULL,

course\_name text NOT NULL,

course\_description text,

Foreign Key(course\_department\_id) references Department(department\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Grading\_Req (

grading\_req\_id serial Primary Key NOT NULL,

grading\_req\_course\_id integer NOT NULL,

grading\_type text NOT NULL,

Foreign Key(grading\_req\_course\_id) references Course(course\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Quiz(

grading\_req\_id serial Primary Key NOT NULL,

quiz\_grade integer NOT NULL,

Foreign Key(grading\_req\_id) references Grading\_Req(grading\_req\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Project(

grading\_req\_id serial Primary Key NOT NULL,

project\_grade integer NOT NULL,

Foreign Key(grading\_req\_id) references Grading\_Req(grading\_req\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Exam(

grading\_req\_id serial Primary Key NOT NULL,

exam\_grade integer NOT NULL,

Foreign Key(grading\_req\_id) references Grading\_Req(grading\_req\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Homework(

grading\_req\_id serial Primary Key NOT NULL,

homework\_grade integer NOT NULL,

Foreign Key(grading\_req\_id) references Grading\_Req(grading\_req\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Private\_File (

private\_user\_id integer NOT NULL,

file\_id integer unique NOT NULL,

file\_name text NOT NULL,

Primary Key(private\_user\_id, file\_id),

Foreign Key(private\_user\_id) references Db\_User(user\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Achievement (

achievement\_id serial Primary Key NOT NULL,

achievement\_type text NOT NULL,

achievement\_date Date

);

CREATE TABLE Willing\_Project (

achievement\_id serial Primary Key NOT NULL,

project\_name text NOT NULL,

Foreign Key(achievement\_id) references Achievement(achievement\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Language (

achievement\_id serial Primary Key NOT NULL,

language\_name text NOT NULL,

Foreign Key(achievement\_id) references Achievement(achievement\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Certificate (

achievement\_id serial Primary Key NOT NULL,

certificate\_name text NOT NULL,

Foreign Key(achievement\_id) references Achievement(achievement\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Test\_Score (

achievement\_id serial Primary Key NOT NULL,

test\_name text NOT NULL,

score integer NOT NULL,

Foreign Key(achievement\_id) references Achievement(achievement\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Collection (

collection\_user\_id integer NOT NULL,

collection\_id serial Primary Key NOT NULL,

Foreign Key(collection\_user\_id) references Db\_User(user\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Job\_Offer (

job\_offer\_id serial Primary Key NOT NULL,

job\_offer\_company\_id integer NOT NULL,

job\_collection\_id integer NOT NULL,

job\_title text NOT NULL,

job\_location text,

Foreign Key(job\_collection\_id) references Collection(collection\_id) ON DELETE CASCADE ON UPDATE CASCADE,

Foreign Key(job\_offer\_company\_id) references Company(company\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Post (

post\_id serial Primary Key NOT NULL,

post\_collection\_id integer NOT NULL,

Foreign Key(post\_collection\_id) references Collection(collection\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE View\_Profile(

user\_profile\_id integer NOT NULL,

viewer\_id integer NOT NULL,

Primary key (user\_profile\_id, viewer\_id),

FOREIGN KEY(user\_profile\_id) REFERENCES User\_Profile (user\_profile\_id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY(viewer\_id) REFERENCES User\_Profile (user\_profile\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Connect(

user\_profile\_id integer NOT NULL,

user\_connect\_id integer NOT NULL,

Primary key (user\_profile\_id, user\_connect\_id),

FOREIGN KEY(user\_profile\_id) REFERENCES User\_Profile (user\_profile\_id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY(user\_connect\_id) REFERENCES User\_Profile (user\_profile\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Enroll(

user\_id integer NOT NULL,

course\_id integer NOT NULL,

grade integer NOT NULL,

Primary Key(user\_id, course\_id),

FOREIGN KEY(user\_id) REFERENCES Db\_User (user\_id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY(course\_id) REFERENCES Course (course\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Teach(

user\_id integer NOT NULL,

course\_id integer NOT NULL,

year integer,

semester text,

Primary Key(user\_id, course\_id),

FOREIGN KEY(user\_id) REFERENCES Db\_User (user\_id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY(course\_id) REFERENCES Course (course\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Achieve(

user\_id integer NOT NULL,

achievement\_id integer NOT NULL,

Primary Key(user\_id, achievement\_id),

FOREIGN KEY(user\_id) REFERENCES Db\_User (user\_id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY(achievement\_id) REFERENCES Achievement (achievement\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Apply(

user\_id integer NOT NULL,

job\_offer\_id integer NOT NULL,

Primary Key (user\_id, job\_offer\_id),

Foreign Key(user\_id) references Db\_User(user\_id) ON DELETE CASCADE ON UPDATE CASCADE,

Foreign Key(job\_offer\_id) references Job\_Offer(job\_offer\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE View(

user\_id integer NOT NULL,

job\_offer\_id integer NOT NULL,

Primary Key(user\_id, job\_offer\_id),

Foreign Key(user\_id) references Db\_User(user\_id) ON DELETE CASCADE ON UPDATE CASCADE,

Foreign Key(job\_offer\_id) references Job\_Offer(job\_offer\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Post\_Share(

post\_id integer NOT NULL,

user\_id integer NOT NULL,

Primary Key(post\_id, user\_id),

Foreign Key(post\_id) references Post(post\_id) ON DELETE CASCADE ON UPDATE CASCADE,

Foreign Key(user\_id) references Db\_User(user\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Post\_Like(

post\_id integer NOT NULL,

user\_id integer NOT NULL,

Primary Key (post\_id, user\_id),

Foreign Key(post\_id) references Post(post\_id) ON DELETE CASCADE ON UPDATE CASCADE,

Foreign Key(user\_id) references Db\_User(user\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE Post\_Comment(

post\_id integer NOT NULL,

user\_id integer NOT NULL,

Primary Key (post\_id, user\_id),

Foreign Key(post\_id) references Post(post\_id) ON DELETE CASCADE ON UPDATE CASCADE,

Foreign Key(user\_id) references Db\_User(user\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

## 4.2 Tuples

INSERT INTO Db\_User VALUES (100000,'worthbee',2154523,True,False,False,null,null,300000,null,null,900000,null,null);

INSERT INTO Db\_User VALUES (100001,'decayvowel',52312,True,True,False,null,'Undergraduate',300000,300001,null,900001,null,null);

INSERT INTO Db\_User VALUES (100002,'pradawreck',54413312,False,False,True,'Advertisement',null,null,null,800000,900002,'Computer Engineer',('01-06-2019',null));

INSERT INTO Db\_User VALUES (100003,'cartierlinear',784515,True,True,True,'Banking','PHD',300000,300002,800001,900003,'Accounter',('16-09-2020','16-11-2020'));

INSERT INTO Db\_User VALUES (100004,'surroundgrind',214566,True,False,True,'Software',300001,null,300002,null,900004,'Human Resources',('29-07-2021',null));

INSERT INTO Db\_User VALUES (100005,'jestawful',88962,True,False,False,null,null,300003,null,null,900005,null,null);

INSERT INTO Db\_User VALUES (100006,'frankfurtercovey',09066,False,True,False,null,'Graduate',null,300001,null,900006,null,null);

INSERT INTO Db\_User VALUES (100007,'spindlereverse',514812,False,False,True,'Hardware',null,null,null,800003,900007,'Software Engineer',('03-01-2021','19-12-2021'));

INSERT INTO Db\_User VALUES (100008,'increasedrash',798900,False,True,True,'Banking','Proffesor',null,300001,800001,900008,'Security',('24-04-2018',null));

INSERT INTO Db\_User VALUES (100009,'sowseforgetful',889521,True,True,True,'Marketing','PHD',300004,300001,800000,900009,'Manager',('19-11-2017','02-09-2020'));

INSERT INTO Db\_User VALUES (100010,'sweepevery',2832521,False,False,True,'Hardware',null,null,null,800001,900010,'Manager',('19-01-2018',null));

INSERT INTO Db\_User VALUES (100011,'founderwonder',5687521,False,False,True,'Banking',null,null,null,800002,900011,'Manager',('12-05-2016',null));

INSERT INTO Db\_User VALUES (100012,'phalangesexcited',5671521,False,False,True,'Marketing',null,null,null,800003,900012,'Manager',('16-06-2015',null));

## 4.3 Triggers

CREATE FUNCTION gpa\_controller()

RETURNS TRIGGER AS $gpa\_controller$

BEGIN

--check that new gpa is above 1,80

IF NEW.gpa < 1.80 THEN

RAISE EXCEPTION 'Transcript GPA can not be below 1,80.';

END IF;

END;

$gpa\_controller$ LANGUAGE plpgsql;

CREATE TRIGGER trigger\_gpa

BEFORE INSERT OR UPDATE OF gpa ON TRANSCRIPT

FOR EACH ROW

EXECUTE FUNCTION gpa\_controller();

CREATE FUNCTION college\_name()

RETURNS TRIGGER AS $college\_name$

BEGIN

--check that college\_name is given

IF NEW.college\_name IS NULL THEN

RAISE EXCEPTION 'College Name cannot be null';

END IF;

END;

$college\_name$ LANGUAGE plpgsql;

CREATE TRIGGER trigger\_college\_name

BEFORE INSERT OR UPDATE OF college\_name ON COLLEGE

FOR EACH ROW

EXECUTE FUNCTION college\_name();

CREATE FUNCTION psswrd\_change()

RETURNS TRIGGER AS $psswrd\_change$

BEGIN

--check that password has changed

IF NEW.psswrd <> OLD.psswrd THEN

RAISE EXCEPTION 'Password changed.';

END IF;

--check that new password is the same as the old one

IF NEW.psswrd = OLD.psswrd THEN

RAISE EXCEPTION 'New password can not be the same as the old one.';

END IF;

END;

$psswrd\_change$ LANGUAGE plpgsql;

CREATE TRIGGER trigger\_psswrd

BEFORE UPDATE OF psswrd ON DB\_USER

FOR EACH ROW

EXECUTE FUNCTION psswrd\_change();

## 4.4 Check Constraints

ALTER TABLE Db\_User

ADD CONSTRAINT Db\_User\_CheckFinishDateGreaterThanStartDate

CHECK ((work\_time).finish\_date >= (work\_time).start\_date) ;

ALTER TABLE Quiz

ADD CONSTRAINT GRADE\_CHECK

CHECK (quiz\_grade>=40);

ALTER TABLE Project

ADD CONSTRAINT GRADE\_CHECK

CHECK (project\_grade>=40);

ALTER TABLE Exam

ADD CONSTRAINT GRADE\_CHECK

CHECK (exam\_grade>=40);

ALTER TABLE Homework

ADD CONSTRAINT GRADE\_CHECK

CHECK (homework\_grade>=40);

ALTER TABLE Achievement

ADD CONSTRAINT TYPE\_CHECK\_ACHIEVEMENT

CHECK (achievement\_type IN ('Willing Project','Language','Test\_Score','Certificate'));

ALTER TABLE Grading\_Req

ADD CONSTRAINT TYPE\_CHECK\_REQ

CHECK (grading\_type IN ('Quiz','Project','Exam','Homework'));

ALTER TABLE User\_Profile

ADD CONSTRAINT CHECK\_SEX

CHECK (Sex IN ('M','F'));

ALTER TABLE Db\_User

ADD CONSTRAINT CHECK\_LENGTH

CHECK (user\_id Between 100000 and 199999);

ALTER TABLE Transcript

ADD CONSTRAINT CHECK\_LENGTH

CHECK (transcript\_no Between 200000 and 299999);

ALTER TABLE College

ADD CONSTRAINT CHECK\_LENGTH

CHECK (college\_id Between 300000 and 399999);

ALTER TABLE Faculty

ADD CONSTRAINT CHECK\_LENGTH

CHECK (faculty\_id Between 400000 and 499999);

ALTER TABLE Department

ADD CONSTRAINT CHECK\_LENGTH

CHECK (department\_id Between 500000 and 599999);

ALTER TABLE Course

ADD CONSTRAINT CHECK\_LENGTH

CHECK (course\_id Between 600000 and 699999);

ALTER TABLE Grading\_Req

ADD CONSTRAINT CHECK\_LENGTH

CHECK (grading\_req\_id Between 700000 and 799999);

ALTER TABLE Company

ADD CONSTRAINT CHECK\_LENGTH

CHECK (company\_id Between 800000 and 899999);

ALTER TABLE User\_Profile

ADD CONSTRAINT CHECK\_LENGTH

CHECK (user\_profile\_id Between 900000 and 999999);

## 4.5 SQL Statements

### 4.5.a INSERT/DELETE/UPDATE Statements

INSERT INTO Company\_Profile VALUES(700000,'ARÇELİK','İZMİR',123665);

INSERT INTO Company\_Profile VALUES(700001,'KOÇ','ANKARA',124785);

INSERT INTO Company\_Profile VALUES(700002,'BEKO','İSTANBUL',96855);

INSERT INTO Company\_Profile VALUES(700003,'ACUN MEDYA','İSTANBUL',127765);

DELETE FROM Db\_User

WHERE username = 'worthbee';

DELETE FROM Faculty

WHERE faculty\_college\_id=2

DELETE FROM Company\_Profile

WHERE company\_location ='İZMİR'

UPDATE Db\_User

SET sector='Marketing'

WHERE user\_id = 100003

UPDATE Company\_Profile

SET company\_location ='ANKARA'

WHERE company\_location='İSTANBUL'

UPDATE Achievement

SET achievement\_date ='03-06-2022'

WHERE achievement\_id=110016

### 4.5.b Select Statements

#### i) One Table

1)

Retrieve the full name, address, sex , birth date and phone of all users.

SELECT full\_name,address,sex,BDate,phone

FROM User\_Profile

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

2)

Retrieve the user id and the user name of users who is either both student or teacher or both student or worker.

SELECT user\_id,username

FROM Db\_User

WHERE student\_flag=true AND teacher\_flag=true

UNION

SELECT user\_id,username

FROM Db\_User

WHERE student\_flag=true AND worker\_flag=true

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

3)

Retrieve the year,semester of teacher with ID of 100008.

SELECT user\_id,Teach.year,semester

FROM Teach

WHERE user\_id=100008

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

#### ii)Minimum Two Table

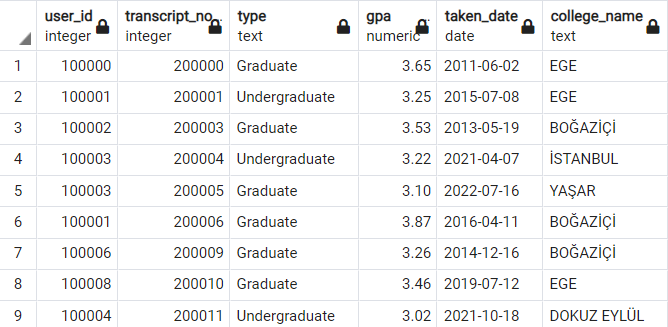
1)

Retrieve the users transcript information who has gpa more than 3.

SELECT user\_id,transcript\_no,type,gpa,taken\_date,college\_name

FROM Db\_User,Transcript,College

WHERE gpa>3 AND t\_user\_id=user\_id AND t\_college\_id=college\_id



2)

Retrieve the career title and company names of user who is worker .

SELECT full\_name,company\_name,career\_title

FROM Db\_User, Company, Company\_Profile, User\_Profile

WHERE worker\_flag=true AND Db\_User.user\_profile\_id=User\_Profile.user\_profile\_id AND worker\_company\_id=company\_id

AND Company.company\_profile\_id=Company\_Profile.company\_profile\_id

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

3)

Retrieve the job offer information which offers by Arçelik.

SELECT job\_title,job\_offer\_id,job\_location,company\_name

FROM Job\_Offer, Company\_Profile,Company

WHERE Company.company\_id=800000 AND Company.company\_profile\_id=Company\_Profile.company\_profile\_id

AND Company.company\_id=Job\_Offer.job\_offer\_company\_id

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

4)

Retrieve the grading requirements of the courses which in the department with ID of 500001

select course\_name, grading\_type

from Course, Grading\_Req, Department

where department\_id=500001 and course\_department\_id=department\_id and course\_id=grading\_req\_course\_id

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

#### iii)Minimum Three Table

1)

Retrieve the transcript information of user with ID of 100000.

select transcript\_no, Transcript.Type, gpa, taken\_date, college\_name

from Db\_User, College, Transcript

where Db\_user.user\_id =100000 and t\_user\_id=user\_id and t\_college\_id=college\_id

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

2)

Retrieve the job offer information which offers by Arçelik.

SELECT job\_title,job\_offer\_id,job\_location,company\_name

FROM Job\_Offer, Company\_Profile,Company

WHERE Company.company\_id=800000 AND Company.company\_profile\_id=Company\_Profile.company\_profile\_id

AND Company.company\_id=Job\_Offer.job\_offer\_company\_id

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

3)

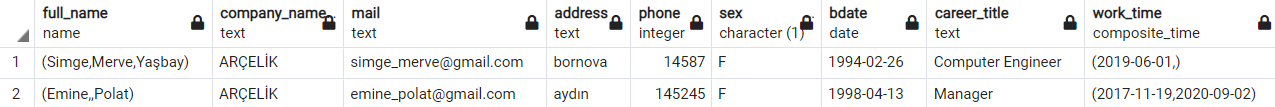
Retrieve the worker's informations who works at the company with id of 800000.

SELECT full\_name,company\_name,mail,address,phone,sex,BDate, career\_title,work\_time

FROM Db\_User,User\_Profile,Company,Company\_Profile

WHERE company\_id=800000 AND Db\_User.worker\_company\_id=company\_id AND Db\_User.user\_profile\_id= User\_Profile.user\_profile\_id

AND Company.company\_profile\_id=Company\_Profile.company\_profile\_id



### 4.5.c Original Select Statements

1)

Retrieve the name of company which located in İzmir or offers a job in İzmir.

select distinct company\_name

from Company\_Profile

where company\_name in (

select company\_name

from Company\_Profile

where company\_location='İZMİR')

or

company\_name in (select company\_name

from Company\_Profile, Job\_Offer, Company

where job\_location='İzmir' and job\_offer\_company\_id=company\_id and Company.company\_profile\_id=Company\_Profile.company\_profile\_id)

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

2)

If the user having 900000 user ids has a connection in the company which has offer a job that the user applied for, it returns that contact's user id.

select user\_id

from Db\_User

where (worker\_company\_id) in (select company\_id

from Apply, Db\_User, Job\_Offer, User\_Profile, Company\_Profile,Company

where User\_Profile.user\_profile\_id=900000 and Db\_User.user\_id=Apply.user\_id

and Apply.job\_offer\_id=Job\_Offer.job\_offer\_id and Job\_Offer.job\_offer\_company\_id=Company.company\_id)

intersect

select user\_id

from Db\_User

where (user\_profile\_id) in (select user\_connect\_id

from Connect, Db\_User, User\_Profile

where User\_Profile.user\_profile\_id=900000 and User\_Profile.user\_profile\_id=Connect.user\_profile\_id)

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

3)

Returns the information of the teacher who teaches the courses in which the student with a user ID of 100000 is registered.

create view teacher(teacher\_id, course\_name, year, semester) as

(select distinct Teach.user\_id, course\_name, Teach.year, semester

from Enroll, Teach, Db\_User, Course, User\_Profile

where Db\_User.user\_id=100000 and Db\_User.user\_id=Enroll.user\_id and Enroll.course\_id=Course.course\_id

and Course.course\_id=Teach.course\_id)

select full\_name, teacher.course\_name, teacher.year, teacher.semester

from teacher, Db\_User, User\_Profile

where teacher.teacher\_id=Db\_User.user\_id and Db\_User.user\_profile\_id=User\_Profile.user\_profile\_id

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

4)

Retrieve the information of courses which has college\_id of 300000

select college\_name, faculty\_name, department\_name, course\_name

from College, Faculty, Department, Course

where College.college\_id=300000 AND College.college\_id = Faculty.faculty\_college\_id AND Faculty.faculty\_id=Department.department\_faculty\_id AND Department.department\_id=Course.course\_department\_id

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

5)

Retrive the course names and faculty names which contains "Engineering" in all colleges.

SELECT course\_name,college\_name,faculty\_name,department\_name

FROM College, Faculty, Department, Course

WHERE department\_name LIKE '%Engineering' AND Department.department\_faculty\_id=Faculty.faculty\_id

AND Faculty.faculty\_college\_id= College.college\_id AND Department.department\_id=Course.course\_department\_id

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu