**EGE UNIVERSITY**

**FACULTY OF ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

****

**DATABASE MANAGEMENT**

**2021-2022 FALL SEMESTER**

**DATABASE MANAGEMENT TERM PROJECT REPORT**

**PREPARED BY**

**Eren Tahir - 05200000036**

**Emine Polat - 051800000041**

**Ahmet Çökmez - 05180000111**

**DELIVERY DATE**

**02.02.2022**

# ANALYSIS

## Brief explanation of LinkedIn

LinkedIn is a social media platform like Facebook or Instagram where users can share posts, make connections and interact with each other. Howerver unlike them, LinkedIn is specifically designed for proffesional business life and ındustry networking. Users can share their proffesional work lives, , can easily find internships and job ads that are suitable for them and can contact with other professional and expand their network to contribute their careers.

## Brief Explanation of Moodle

Moodle is an online course management system. There may be individual instructors who teach courses to few students, as well as universities that provide education to thousands of students. Moodle can be used to maintain distance education or to contribute to face-to-face education. With Moodle, instructors can share course resources with students, give exams for tracking students’ progress, can receive assignments, can answer students' questions in the forum area.

## 2.1 Analysis Report For LinkedIn

### Aim of LinkedIn:

LinkedIn intend to bring people in business life together and to help them to communicate with each other, be aware of suitable job and internship opportunities, reach companies easily. And create an environment for companies and individuals to introduce themselves to other users in a professional way. Briefly purpose of LinkedIn is to bring together professionals from all over the world and help them advance in their careers.

### Main Entities of LinkedIn:

* MEMBER
* MEMBER\_PROFİLE
* POSTS
* CONNECTIONS
* ORGANIZATIONS
* CV

### Characteristics of Each entity:

**MEMBER :**

* member\_id
* organization\_id
* profile\_id
* email
* member\_password
* first\_name
* last\_name
* m\_name
* gender
* current\_job\_info
* cv\_id
* birthday

**MEMBER\_PROFILE**

* profile\_id
* member\_id
* date\_created
* date\_last\_updated

**POST**

* post\_id
* member\_id
* post\_type
* post\_context

**CONNECTIONS**

* connection\_id
* connection\_member\_id
* connected\_member\_id
* connection\_made\_date

**ORGANIZATIONS**

* organization\_id
* organization\_name
* organization\_ındustry
* organization\_location

**CV**

* cv\_id
* member\_id
* date\_created
* date\_updated
* cv\_context

### Relationships in LinkedIn:

* MEMBER **HAS** MEMBER\_PROFILE
* MEMBER **HAS** CV
* MEMBER **BELONG** ORGANIZATIONS
* MEMBER **MAKE** CONNECTION
* MEMBER **SHARE** POST
* MEMBER **LIKE** POST
* MEMBER **COMMENT** POST
* MEMBER **MESSAGE** MEMBER
* CONNECTION **HAS** MEMBER

### Constraints, Caracteristics, Relationships Among Entities of LinkedIn :

In the LinkedIn network, members can establish unlimited connections with other members. Members must have only one profile. A profile must belong to only one member. A LinkedIn member can establish one or more organizations and other members can join these organizations. Organizations must have a founding member. Members in the Linkedin network may not post at all or may post countless. Members can like or make comment to the posts. A post must be shared by a member. Members can send message to other members. Members can add their cv information to the Linkedin network. A member can have one or more CVs.

## 2.2 Analysis Report For Moodle

### Aim of Moodle:

The main purpose of Moodle is to provide an educational environment that is easily accesible,useble and reliable for contributing to the development of insturctors and students,facilitating course management, regardless of time and place.

### Main Entities of Moodle:

* ADMIN
* INSTRUCTOR
* STUDENT
* COURSE
* WEEK
* DOCUMENT
* EXAM
* LIVESESSION
* ASSIGNMENT

### Characteristics of Each entity:

Just necessary attributes are handled. Because of the fact that Moodle is very customizable that results in attributes and relationships to vary in different ways. In the big part of the analysis, EgeDers had been considered.

**ADMIN:**

* user\_id
* email
* password
* first\_name
* last\_name

**INSTRUCTOR:**

* user\_id
* email
* password
* first\_name
* last\_name

**STUDENT:**

* user\_id
* email
* password
* first\_name
* last\_name

**COURSE:**

* course\_code
* section
* semester
* course\_name
* join\_password(nullable)

**WEEK:**

* week\_id
* title
* text
* course\_code
* section
* semester

**DOCUMENT:**

* file\_path
* title
* course\_code
* section
* semester

**EXAM:**

* exam\_id
* title
* starting\_time
* ending\_time
* explanation\_content

**LIVE\_SESSION:**

* connection\_link
* title
* starting\_time
* ending\_time

**ASSIGNMENT:**

* assignment\_id
* title
* ending\_time
* explanation\_content

### Relationships in Moodle:

* ADMIN **MANAGE** SYSTEM
* ADMIN **CREATE** COURSE
* ADMIN **APPEND** USER **to** COURSE
* ADMIN **ASSIGN** INSTRUCTOR **to** COURSE
* COURSE **HAS** WEEK
* WEEK **HAS** ASSIGMENT
* WEEK **HAS** DOCUMENT
* WEEK **HAS** EXAM
* WEEK **HAS** LIVE\_SESSION
* STUDENT **JOIN** LIVE\_SESSION
* STUDENT **UPLOAD** DOCUMENT **to** EXAM
* STUDENT **UPLOAD** DOCUMENT **to** ASSIGMENT
* INSTRUCTOR **GRADE** ASSIGMENT
* INSTRUCTOR **GRADE** EXAM
* INSTRUCTOR **START** LIVE\_SESSION

### Constraints, Caracteristics, Relationships Among Entities of Moodle:

Education system is administered by admins. A system must have at least one admin. Admins can append users of all types (admin,assistant, instructor, student) into the system. Admins create courses and assign instructors, his/her assistants (instance of Student entity) and students. A course must have only one instructor. Admins are authorized with all rights, control everything in the system. Courses are divided into weeks and managed by the instructor and his/her assistants.Week part can include documents, exams, assignments and live sessions. A live session must be given by just one tutor(assistant or instructor). Managing users of courses add these entities into weeks. Students can download documents, examine weeks, in addition they are expected to join live sessions, attend exams and do assignments in order to pass courses. Students can upload documents to exams and assignments. Exams and assignments are graded by managing users of courses. Live sessions are given by managing users of courses.

# CONCEPTUAL DESIGN

### Important Point of Our Design:

The main goal of the project is to catch and imply common characteristics of Moodle and Linkedin. Firstly, Moodle is a course administration system and Linkedin is a platform that is a mixture of a social website and a business website,in addition neither creating a company nor an university database is the goal of the project so we didn’t handle hierarchies. Secondly, business and education fields intercept at universities and companies, not at hobby or any other non professional courses.Thus, there will be two organization types; university and company. We realized companies educate their staff, therefore we compromised on the “companies can give courses'' idea and put the idea in our project. A user can use every social feature like sending messages to other users,creating or joining a society, posting contents and interacting with other contents. A user can administer a company as an admin. An admin’s role is just to administrate the organization that he is part of. A standard user can be a member of different organizations, so that there is a need for authorization to create and give courses, administer rights on each organization that the one is a member of. All other features of Linkedin and Moodle are aimed to be implied on the project.

### Data Requirements:

The system has only one type of user. Each user has an unique username, id, and email. Each user also have first name, last name, sex, birthday, country and creation time of the profile. User can upload only one CV. A CV can contain multiple skills, cerficates, work experience and education history. Each CV has an unique id. Each CV also have a description which is bio and have the time of last updated. An user can open more than one course or attend more than one course. And also an user can manage more than one organisation’s profile and can create new admins for the organisation which he/she manages. An user can be a part of several organizations, can follow and message other users. Users can like, comment or share posts and can create or apply for job adverts.

The database keeps track of three types of organisations: company, university and society. An organisation must belong to only one of them. Each organisation has an unique name, id, email, telephone and website link. Each organisation also has a location which contains street, city and country information. An organisation can have more than one admin. Admin is a person who manages the organisation profile. There may be more than one person who is in an organisation. An organisation can share a post or a job ad. Every company has industry information unlike the university and society.

The database should track the courses. Each course has an unique id, a name, a password for attending the course, start date and end date. Each course must have a tutor. Tutors can add assistants to the course. A course can have more than one student. A course consists of sections. The Tutor and assistans can load contents to the sections and can open assignments. Also they can start a live session. Live sessions and assignments have a start time and an end time. Students can attend live sessions, upload content for the assignments.

**RELATIONS:**

**USER**

* 0,N open and manage multiple groups.
* 0,N may belong to the organization.
* 0,N can post job advert.
* 0,N can send or receive messages.
* 1,1 must have one and only one CV.
* 0,N can share post.
* 0,N can apply for job advert.
* 0,N can follow users.
* 0,N belong to courses
* 0,N give courses
* 0,N can like or comment on post.

**ORGANIZATION**

* 1,1 must created by one and only one user.
* 1,N can managed by users.
* 1,N may have users.
* 0,N can post job advert.
* 0,N can send or receive messages.
* 0,N can share post.
* 0,N can like or comment on post
* 0,N give courses

**COMPANY**

* 1,1 must created by one and only one user.
* 1,N can managed by users.
* 1,N may have users.
* 0,N can post job advert.
* 0,N can send or receive messages.
* 0,N can share post.
* 0,N can like or comment on post
* 0,N give courses

**UNIVERSITY**

* 1,1 must created by one and only one user.
* 1,N can managed by users.
* 1,N may have users.
* 0,N can post job advert.
* 0,N can send or receive messages.
* 0,N can share post.
* 0,N can like or comment on post
* 0,N give courses

**SOCIETY**

* 1,1 must created by one and only one user.
* 1,N can managed by users.
* 1,N may have users.
* 0,N can post job advert.
* 0,N can send or receive messages.
* 0,N can share post.
* 0,N can like or comment on post
* 0,N give courses

**COURSE**

* 1,1 must created and managed by one and only one user.
* 1,N may have users.
* 1,N may have sections.
* 0,N can have assistant.

**SECTION**

* 0,N can have live sessions.
* 1,1 must depends on one and only one course.
* 0,N can have assignments.

**LIVE\_SESSION**

* 1,1 belong to section

**ASSIGNMENT**

* 0,N users can upload homework to it
* 1,1 belong to section

**POST**

* 1,1 must posted by one and only one user,organization,society or company
* 0,N can have comments or likes.

**JOB\_ADVERT**

* 1,1 must opened by one and only one user,organization,society or company
* 1,N users can apply to job adverts.

**MESSAGE**

* 1,1 sended by user,organization,society or company
* 1,1 recieved by user,organization,society or company

**CV**

* 1,1 must belong to one and only one user.
* 0,N may have job history.
* 0,N may have certificate.
* 0,N may have skill.
* 0,N may have education.

**JOB\_HISTORY**

* 1,1 must belong to one and only one CV.

**Education**

* 1,1 must belong to one and only one CV.

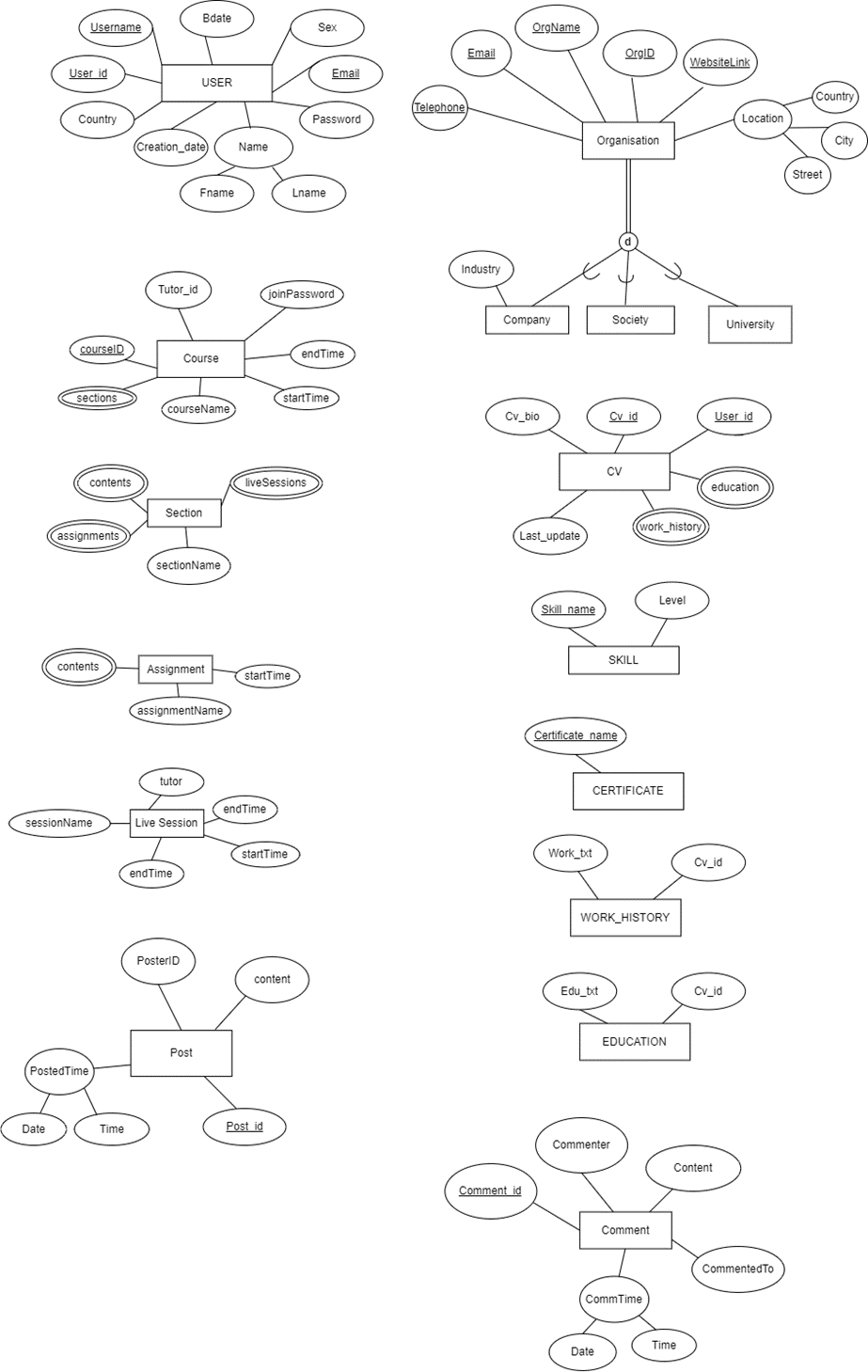
**CERTIFICATE**

* 0,N belong to CV.

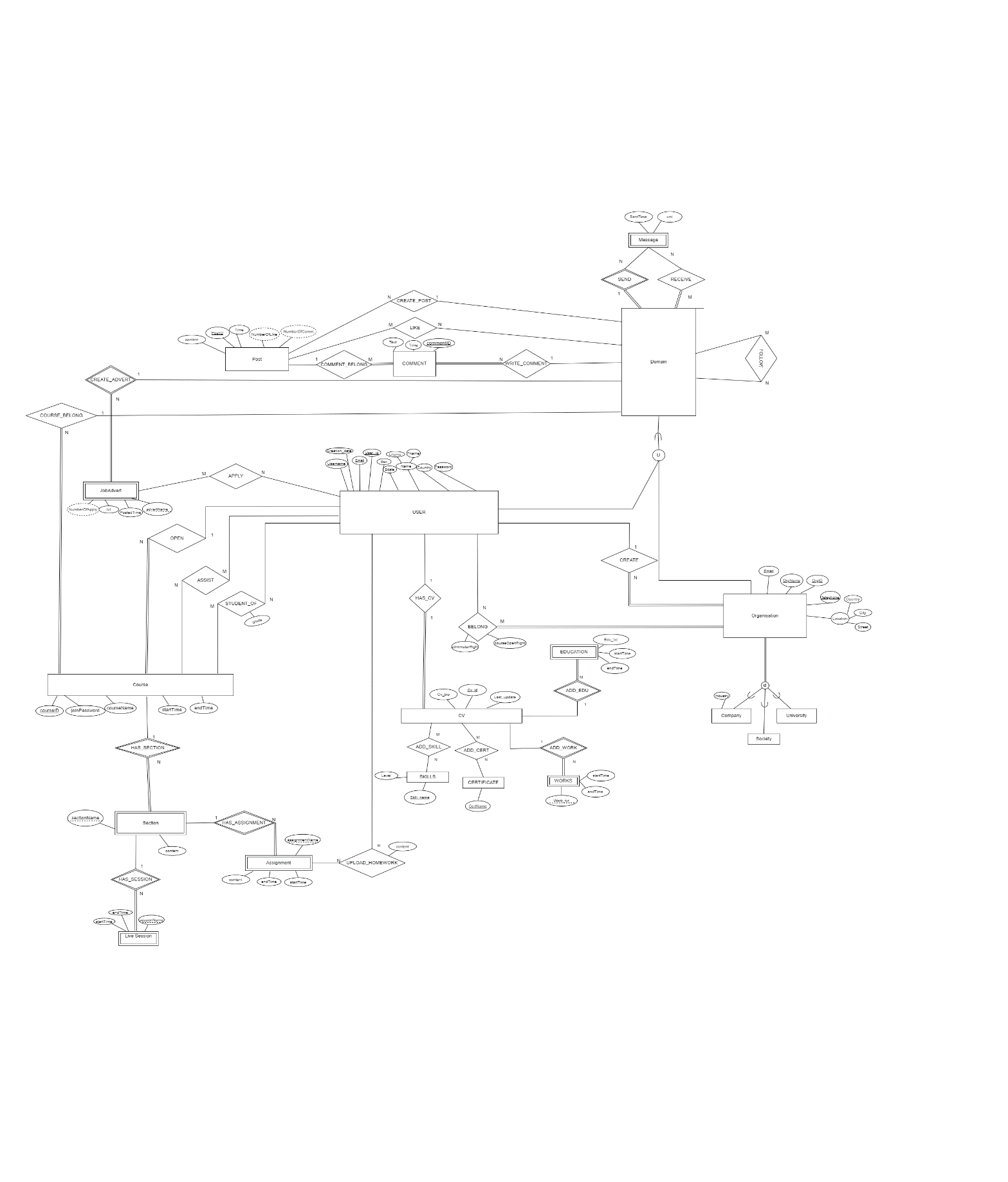
**SKILL**

* 0,N belong to CV.

### Initial Design



### EER Diagram:



# LOGICAL MODEL

**1.ITERATION**

**STEP1:**

* COURSE (CourseId,CourseName,JoınPassword,StartTime,EndTime)
* CV (CvId,CvBio,LastUpdate)
* SKILL (SkillName, Level)
* CERTIFICATE (CertName)
* POST (PostId, Content, Time)
* COMMENT( CommentID, Text, Time)

**STEP2:**

* SECTION (CourseId,SectionName, Content)
* EDUCATION (CvId,EduName, StartTime, EndTime)
* JOB\_HISTORY (CvId,JobName, StartTime, EndTime)

**STEP3:**

We don’t have regular one to one relations.

**STEP4:**

We don’t have regular one to many relations.

**STEP5:**

* ADD\_SKILL(CvId,SkillName)
* ADD\_CERT(CvId,CertName)

**STEP6:**

We don’t have multivalued attributes.

**STEP7:**

We don’t have N-ary relations.

**STEP8:**

**STEP9:**

* USER(UserId, Email, Username, Password, CreationDate, Sex, Bdate, Fname, Lname, Country, DomainName)
* ORGANIZATION(OrgId,OrgName,Email,Telephone, Country, City, Street, DomainName)
* DOMAIN(DomainName)

**2.ITERATION**

**STEP1:**

**STEP2:**

* JOB\_ADVERT(DomainName,AdvertName, PostedTime, Content)
* ASSIGNMENT(CourseId,SectionName,AssıgnmentName, StartTime, EndTime, Content)
* LIVE\_SESSION(CourseId,SectionName,SessionName, StartTime, EndTime)

**STEP3:**

* CV(CvId, CvBio, LastUpdate, UserId)

**STEP4:**

*via CREATE realtionship*

* ORGANIZATION(OrgId, OrgName ,Email, Telephone, Country, City, Street, DomainName, UserId)

*via CREATE\_POST relationship*

* POST(PostId, Content, Time, DomainName)

*via OPEN relationship*

* COURSE(CourseId, CourseName, JoınPassword, StartTime, EndTime, UserId)

*via COURSE\_BELONG relationship*

* COURSE(CourseId, CourseName, JoınPassword, StartTime, EndTime ,UserId, DomainName)

*via WRITE\_COMMENT relationship*

* COMMENT(CommentID, Time, Text, DomainName)

*via COMMENT\_BELONG relationship*

* COMMENT(CommentID, Time, Text, DomainName, PostID)

**STEP5 :**

* LIKE (DomainName,PostID)
* FOLLOW (Follower,Following)
* MESSAGE (Sender,Receiver, SendTime, Content)
* APPLY (UserId,DomainName,AdvertName)
* ASSIST (UserId,CourseId)
* STUDENT\_OF (UserId,CourseId, Grade)
* UPLOAD\_HOMEWORK (UserId,CourseId,SectionName,AssıgnmentName,Content)
* BELONG (UserId,OrgId, CourseOpenRight, AdministerRight)

**STEP6:**

**STEP7:**

**STEP8:**

*via 8-b method*

* COMPANY(OrgId, Industry)
* UNIVERSITY(OrgId)
* SOCIETY(OrgId)

**STEP 9:**

**COMPANY**

|  |  |
| --- | --- |
| OrgId | Industry |

**UNIVERSITY**

|  |
| --- |
| OrgId |

**SOCIETY**

|  |
| --- |
| OrgId |

**BELONG**

|  |  |  |  |
| --- | --- | --- | --- |
| UserId | OrgId | CourseOpenRight | AdministerRight |

**UPLOAD\_HOMEWORK**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UserId | CourseId | SectionName | AssıgnmentName | Content |

**STUDENT\_OF**

|  |  |  |
| --- | --- | --- |
| UserId | CourseId | Grade |

**ASSIST**

|  |  |
| --- | --- |
| UserId | CourseId |

**APPLY**

|  |  |  |
| --- | --- | --- |
| UserId | DomainName | AdvertName |

**MESSAGE**

|  |  |  |  |
| --- | --- | --- | --- |
| Sender | Receiver | SendTime | Content |

**FOLLOW**

|  |  |
| --- | --- |
| Follower | Following |

**LIKE**

|  |  |
| --- | --- |
| DomainName | PostID |

**COMMENT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CommentID | Time | Text | DomainName | PostID |

**COURSE**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CourseId | CourseName | JoınPassword | StartTime | EndTime | UserId | DomainName |

**POST**

|  |  |  |  |
| --- | --- | --- | --- |
| PostId | Content | Time | DomainName |

**ORGANIZATION**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| OrgId | OrgName | Email | Telephone | Country | City | Street | DomainName | UserId) |

**CV**

|  |  |  |  |
| --- | --- | --- | --- |
| CvId | CvBio | LastUpdate | UserId |

**LIVE\_SESSION**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CourseId | SectionName | SessionName | StartTime | EndTime |

**ASSIGNMENT**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CourseId | SectionName | AssıgnmentName | StartTime | EndTime | Content |

**JOB\_ADVERT**

|  |  |  |  |
| --- | --- | --- | --- |
| DomainName | AdvertName | PostedTime | Content |

**DOMAIN**

|  |
| --- |
| DomainName |

**USER**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| UserId | Email | Username | Password | CreationDate | Sex | Bdate | Fname | Lname | Country | DomainName |

**ADD\_CERT**

|  |  |
| --- | --- |
| CvId | CertName |

**ADD\_SKILL**

|  |  |
| --- | --- |
| CvId | SkillName |

**JOB\_HISTORY**

|  |  |  |  |
| --- | --- | --- | --- |
| CvId | JobName | StartTime | EndTime |

**EDUCATION**

|  |  |  |  |
| --- | --- | --- | --- |
| CvId | EduName | StartTime | EndTime |

**SECTION**

|  |  |  |
| --- | --- | --- |
| CourseId | SectionName | Content |

**SKILL**

|  |  |
| --- | --- |
| SkillName | Level |

**CERTIFICATE**

|  |
| --- |
| CertName |

# IMPLEMENTATION-PHYSICAL MODEL

### SQL scripts (DDL statements) for creating the database:

CREATE TABLE IF NOT EXISTS public."DOMAIN"

(

    "DomainName" character varying(100) COLLATE pg\_catalog."default" NOT NULL,

    CONSTRAINT "DOMAIN\_pkey" PRIMARY KEY ("DomainName")

)

CREATE TABLE IF NOT EXISTS public."USER"

(

    "UserId" integer NOT NULL,

    "Email" character varying(100) COLLATE pg\_catalog."default" NOT NULL,

    "Username" character varying(45) COLLATE pg\_catalog."default" NOT NULL,

    "Password" character varying(20) COLLATE pg\_catalog."default" NOT NULL,

    "Sex" character varying(25) COLLATE pg\_catalog."default",

    "Fname" character varying(45) COLLATE pg\_catalog."default" NOT NULL,

    "Lname" character varying(45) COLLATE pg\_catalog."default" NOT NULL,

    "Country" character varying(45) COLLATE pg\_catalog."default",

    "DomainName" character varying(100) COLLATE pg\_catalog."default" NOT NULL,

    "CreationDate" character varying COLLATE pg\_catalog."default",

    "Bdate" character varying COLLATE pg\_catalog."default",

    CONSTRAINT "USER\_pkey" PRIMARY KEY ("UserId"),

    CONSTRAINT "Username\_Unique" UNIQUE ("Username"),

    CONSTRAINT "USER\_FK" FOREIGN KEY ("DomainName")

        REFERENCES public."DOMAIN" ("DomainName") MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

)

CREATE TABLE IF NOT EXISTS public."COURSE"

(

    "CourseId" integer NOT NULL,

    "CourseName" character varying(45) COLLATE pg\_catalog."default" NOT NULL,

    "JoinPassword" integer NOT NULL,

    "StartTime" timestamp without time zone NOT NULL DEFAULT CURRENT\_TIMESTAMP,

    "EndTime" timestamp without time zone,

    "UserId" integer NOT NULL,

    "DomainName" character varying(100) COLLATE pg\_catalog."default" NOT NULL,

    CONSTRAINT "COURSE\_pkey" PRIMARY KEY ("CourseId"),

    CONSTRAINT "COURSE\_FK1" FOREIGN KEY ("DomainName")

        REFERENCES public."DOMAIN" ("DomainName") MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION,

    CONSTRAINT "COURSE\_FK2" FOREIGN KEY ("UserId")

        REFERENCES public."USER" ("UserId") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE

)

CREATE TABLE IF NOT EXISTS public."SECTION"

(

    "CourseId" integer NOT NULL,

    "SectionName" character varying(45) COLLATE pg\_catalog."default" NOT NULL,

    "Content" xml,

    CONSTRAINT "SECTION\_pkey" PRIMARY KEY ("SectionName", "CourseId"),

    CONSTRAINT "SectionName\_Unique" UNIQUE ("SectionName"),

    CONSTRAINT "SECTION\_FK" FOREIGN KEY ("CourseId")

        REFERENCES public."COURSE" ("CourseId") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE

)

CREATE TABLE IF NOT EXISTS public."LIVE\_SESSION"

(

    "CourseId" integer NOT NULL,

    "SectionName" character varying(45) COLLATE pg\_catalog."default" NOT NULL,

    "SessionName" character varying(45) COLLATE pg\_catalog."default" NOT NULL,

    "StartTime" timestamp without time zone NOT NULL,

    "EndTime" timestamp without time zone NOT NULL,

    CONSTRAINT "LIVE\_SESSION\_pkey" PRIMARY KEY ("CourseId", "SectionName", "SessionName"),

    CONSTRAINT "LIVE\_SESSION\_FK1" FOREIGN KEY ("CourseId")

        REFERENCES public."COURSE" ("CourseId") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE

        NOT VALID,

    CONSTRAINT "LIVE\_SESSION\_FK2" FOREIGN KEY ("SectionName")

        REFERENCES public."SECTION" ("SectionName") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE

)

CREATE TABLE IF NOT EXISTS public."ASSIGNMENT"

(

    "CourseId" integer NOT NULL,

    "SectionName" character varying(45) COLLATE pg\_catalog."default" NOT NULL,

    "AssıgnmentName" character varying(45) COLLATE pg\_catalog."default" NOT NULL,

    "StartTime" timestamp without time zone NOT NULL,

    "EndTime" timestamp without time zone NOT NULL,

    "Content" xml NOT NULL,

    CONSTRAINT "ASSIGNMENT\_pkey" PRIMARY KEY ("CourseId", "SectionName", "AssıgnmentName"),

    CONSTRAINT "AssıgnmentName\_Unique" UNIQUE ("AssıgnmentName"),

    CONSTRAINT "ASSIGNMENT\_FK1" FOREIGN KEY ("CourseId")

        REFERENCES public."COURSE" ("CourseId") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE,

    CONSTRAINT "ASSIGNMENT\_FK2" FOREIGN KEY ("SectionName")

        REFERENCES public."SECTION" ("SectionName") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE

)

CREATE TABLE IF NOT EXISTS public."ASSIST"

(

    "UserId" integer NOT NULL,

    "CourseId" integer NOT NULL,

    CONSTRAINT "ASSIST\_pkey" PRIMARY KEY ("UserId", "CourseId"),

    CONSTRAINT "ASSIST\_FK1" FOREIGN KEY ("UserId")

        REFERENCES public."USER" ("UserId") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE,

    CONSTRAINT "ASSIST\_FK2" FOREIGN KEY ("CourseId")

        REFERENCES public."COURSE" ("CourseId") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE

)

CREATE TABLE IF NOT EXISTS public."UPLOAD\_HOMEWORK"

(

    "UserId" integer NOT NULL,

    "CourseId" integer NOT NULL,

    "SectionName" character varying(45) COLLATE pg\_catalog."default" NOT NULL,

    "AssignmentName" character varying(45) COLLATE pg\_catalog."default" NOT NULL,

    "Content" xml NOT NULL,

    CONSTRAINT "UPLOAD\_HOMEWORK\_pkey" PRIMARY KEY ("AssignmentName", "SectionName", "CourseId", "UserId"),

    CONSTRAINT "UPLOAD\_HOMEWORK\_FK1" FOREIGN KEY ("UserId")

        REFERENCES public."USER" ("UserId") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE,

    CONSTRAINT "UPLOAD\_HOMEWORK\_FK2" FOREIGN KEY ("CourseId")

        REFERENCES public."COURSE" ("CourseId") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE,

    CONSTRAINT "UPLOAD\_HOMEWORK\_FK3" FOREIGN KEY ("SectionName")

        REFERENCES public."SECTION" ("SectionName") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE,

    CONSTRAINT "UPLOAD\_HOMEWORK\_FK4" FOREIGN KEY ("AssignmentName")

        REFERENCES public."ASSIGNMENT" ("AssıgnmentName") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE

)

CREATE TABLE IF NOT EXISTS public."STUDENT\_OF"

(

    "UserId" integer NOT NULL,

    "CourseId" integer NOT NULL,

    "Grade" integer NOT NULL,

    CONSTRAINT "STUDENT\_OF\_pkey" PRIMARY KEY ("UserId", "CourseId"),

    CONSTRAINT "STUDENT\_OF\_FK1" FOREIGN KEY ("UserId")

        REFERENCES public."USER" ("UserId") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE,

    CONSTRAINT "STUDENT\_OF\_FK2" FOREIGN KEY ("CourseId")

        REFERENCES public."COURSE" ("CourseId") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE

)

CREATE TABLE IF NOT EXISTS public."BELONG"

(

    "UserId" integer NOT NULL,

    "OrgId" integer NOT NULL,

    "CourseOpenRight" boolean NOT NULL,

    "AdministerRight" boolean NOT NULL,

    CONSTRAINT "BELONG\_pkey" PRIMARY KEY ("OrgId", "UserId"),

    CONSTRAINT "BELONG\_FK1" FOREIGN KEY ("UserId")

        REFERENCES public."USER" ("UserId") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE,

    CONSTRAINT "BELONG\_FK2" FOREIGN KEY ("OrgId")

        REFERENCES public."ORGANIZATION" ("OrgId") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE

)

CREATE TABLE IF NOT EXISTS public."CV"

(

    "CvId" integer NOT NULL,

    "CvBio" character varying(100) COLLATE pg\_catalog."default",

    "UserId" integer NOT NULL,

    "LastUpdate" character varying COLLATE pg\_catalog."default",

    CONSTRAINT "CvId\_Unique" PRIMARY KEY ("CvId"),

    CONSTRAINT "CV\_FK" FOREIGN KEY ("UserId")

        REFERENCES public."USER" ("UserId") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE

)

CREATE TABLE IF NOT EXISTS public."SKILL"

(

    "SkillName" character varying(45) COLLATE pg\_catalog."default" NOT NULL,

    "SkillType" character varying(45) COLLATE pg\_catalog."default" NOT NULL,

    CONSTRAINT "SKILL\_pkey" PRIMARY KEY ("SkillName")

)

CREATE TABLE IF NOT EXISTS public."CERTIFICATE"

(

    "CertName" character varying(100) COLLATE pg\_catalog."default" NOT NULL,

    CONSTRAINT "CERTIFICATE\_pkey" PRIMARY KEY ("CertName")

)

CREATE TABLE IF NOT EXISTS public."ADD\_SKILL"

(

    "CvId" integer NOT NULL,

    "SkillName" character varying(45) COLLATE pg\_catalog."default" NOT NULL,

    CONSTRAINT "ADD\_SKILL\_pkey" PRIMARY KEY ("CvId", "SkillName"),

    CONSTRAINT "ADD\_SKILL\_FK1" FOREIGN KEY ("CvId")

        REFERENCES public."CV" ("CvId") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE

        NOT VALID,

    CONSTRAINT "ADD\_SKILL\_FK2" FOREIGN KEY ("SkillName")

        REFERENCES public."SKILL" ("SkillName") MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

        NOT VALID

)

CREATE TABLE IF NOT EXISTS public."ADD\_CERT"

(

    "CvId" integer NOT NULL,

    "CertName" character varying(100) COLLATE pg\_catalog."default" NOT NULL,

    CONSTRAINT "ADD\_CERT\_pkey" PRIMARY KEY ("CvId", "CertName"),

    CONSTRAINT "ADD\_CERT\_FK1" FOREIGN KEY ("CvId")

        REFERENCES public."CV" ("CvId") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE

        NOT VALID,

    CONSTRAINT "ADD\_CERT\_FK2" FOREIGN KEY ("CertName")

        REFERENCES public."CERTIFICATE" ("CertName") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE

        NOT VALID

)

CREATE TABLE IF NOT EXISTS public."EDUCATION"

(

    "CvId" integer NOT NULL,

    "EduName" character varying(100) COLLATE pg\_catalog."default" NOT NULL,

    "StartTime" character varying(20) COLLATE pg\_catalog."default" NOT NULL,

    "EndTime" character varying(20) COLLATE pg\_catalog."default" NOT NULL,

    CONSTRAINT "EDUCATION\_pkey" PRIMARY KEY ("CvId", "EduName"),

    CONSTRAINT "EDUCATION\_FK" FOREIGN KEY ("CvId")

        REFERENCES public."CV" ("CvId") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE

        NOT VALID

)

CREATE TABLE IF NOT EXISTS public."JOB\_HISTORY"

(

    "CvId" integer NOT NULL,

    "JobName" character varying(100) COLLATE pg\_catalog."default" NOT NULL,

    "StartTime" date NOT NULL,

    "EndTime" date,

    CONSTRAINT "JOB\_HISTORY\_pkey" PRIMARY KEY ("CvId", "JobName"),

    CONSTRAINT "JOB\_HISTORY\_FK" FOREIGN KEY ("CvId")

        REFERENCES public."CV" ("CvId") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE

)

CREATE TABLE IF NOT EXISTS public."ORGANIZATION"

(

    "OrgId" integer NOT NULL,

    "OrgName" character varying(100) COLLATE pg\_catalog."default" NOT NULL,

    "Email" character varying(45) COLLATE pg\_catalog."default",

    "Telephone" character varying(20) COLLATE pg\_catalog."default",

    "Country" character varying(45) COLLATE pg\_catalog."default",

    "City" character varying(45) COLLATE pg\_catalog."default",

    "Street" character varying(45) COLLATE pg\_catalog."default",

    "DomainName" character varying(100) COLLATE pg\_catalog."default",

    "UserId" integer NOT NULL,

    CONSTRAINT "OrgId\_Unique" PRIMARY KEY ("OrgId"),

    CONSTRAINT "ORG\_FK1" FOREIGN KEY ("DomainName")

        REFERENCES public."DOMAIN" ("DomainName") MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

        NOT VALID,

    CONSTRAINT "ORG\_FK2" FOREIGN KEY ("UserId")

        REFERENCES public."USER" ("UserId") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE

        NOT VALID

)

CREATE TABLE IF NOT EXISTS public."SOCIETY"

(

    "OrgId" integer NOT NULL,

    CONSTRAINT "SOCIETY\_pkey" PRIMARY KEY ("OrgId"),

    CONSTRAINT "SOCIETY\_FK" FOREIGN KEY ("OrgId")

        REFERENCES public."ORGANIZATION" ("OrgId") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE

)

CREATE TABLE IF NOT EXISTS public."COMPANY"

(

    "OrgId" integer NOT NULL,

    "Industry" character varying(45) COLLATE pg\_catalog."default" NOT NULL,

    CONSTRAINT "COMPANY\_pkey" PRIMARY KEY ("OrgId"),

    CONSTRAINT "COMPANY\_FK" FOREIGN KEY ("OrgId")

        REFERENCES public."ORGANIZATION" ("OrgId") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE

)

CREATE TABLE IF NOT EXISTS public."UNIVERSITY"

(

    "OrgId" integer NOT NULL,

    CONSTRAINT "UNIVERSITY\_pkey" PRIMARY KEY ("OrgId"),

    CONSTRAINT "UNIVERSITY\_FK" FOREIGN KEY ("OrgId")

        REFERENCES public."ORGANIZATION" ("OrgId") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE

)

CREATE TABLE IF NOT EXISTS public."JOB\_ADVERT"

(

    "DomainName" character varying(100) COLLATE pg\_catalog."default" NOT NULL,

    "AdvertName" character varying(100) COLLATE pg\_catalog."default" NOT NULL,

    "PostedTime" timestamp without time zone NOT NULL,

    "Content" xml NOT NULL,

    CONSTRAINT "JOB\_ADVERT\_pkey" PRIMARY KEY ("DomainName", "AdvertName"),

    CONSTRAINT "AdvertName\_Unique" UNIQUE ("AdvertName"),

    CONSTRAINT "JOB\_ADVERT\_FK" FOREIGN KEY ("DomainName")

        REFERENCES public."DOMAIN" ("DomainName") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE

        NOT VALID

)

CREATE TABLE IF NOT EXISTS public."APPLY"

(

    "UserId" integer NOT NULL,

    "DomainName" character varying(100) COLLATE pg\_catalog."default" NOT NULL,

    "AdvertName" character varying(100) COLLATE pg\_catalog."default" NOT NULL,

    CONSTRAINT "APPLY\_pkey" PRIMARY KEY ("AdvertName", "DomainName", "UserId"),

    CONSTRAINT "APPLY\_FK1" FOREIGN KEY ("UserId")

        REFERENCES public."USER" ("UserId") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE

        NOT VALID,

    CONSTRAINT "APPLY\_FK2" FOREIGN KEY ("DomainName")

        REFERENCES public."DOMAIN" ("DomainName") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE

        NOT VALID,

    CONSTRAINT "APPLY\_FK3" FOREIGN KEY ("AdvertName")

        REFERENCES public."JOB\_ADVERT" ("AdvertName") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE

        NOT VALID

)

CREATE TABLE IF NOT EXISTS public."MESSAGE"

(

    "Sender" character varying COLLATE pg\_catalog."default" NOT NULL,

    "Receiver" character varying COLLATE pg\_catalog."default" NOT NULL,

    "SendTime" timestamp without time zone NOT NULL DEFAULT CURRENT\_TIMESTAMP,

    "Content" xml NOT NULL,

    CONSTRAINT "MESSAGE\_pkey" PRIMARY KEY ("Sender", "Receiver"),

    CONSTRAINT "MESSAGE\_FK1" FOREIGN KEY ("Sender")

        REFERENCES public."DOMAIN" ("DomainName") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE,

    CONSTRAINT "MESSAGE\_FK2" FOREIGN KEY ("Receiver")

        REFERENCES public."DOMAIN" ("DomainName") MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

)

CREATE TABLE IF NOT EXISTS public."POST"

(

    "PostId" integer NOT NULL,

    "Content" xml NOT NULL,

    "Time" timestamp without time zone NOT NULL,

    "DomainName" character varying(100) COLLATE pg\_catalog."default" NOT NULL,

    CONSTRAINT "PostId\_Unique" PRIMARY KEY ("PostId"),

    CONSTRAINT "POST\_FK" FOREIGN KEY ("DomainName")

        REFERENCES public."DOMAIN" ("DomainName") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE

        NOT VALID

)

CREATE TABLE IF NOT EXISTS public."COMMENT"

(

    "DomainName" character varying COLLATE pg\_catalog."default" NOT NULL,

    "CommentId" integer NOT NULL,

    "PostId" integer NOT NULL,

    "Content" character varying COLLATE pg\_catalog."default" NOT NULL,

    "Time" timestamp without time zone NOT NULL,

    CONSTRAINT "COMMENT\_PK" PRIMARY KEY ("CommentId"),

    CONSTRAINT "COMMENT\_FK1" FOREIGN KEY ("DomainName")

        REFERENCES public."DOMAIN" ("DomainName") MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION,

    CONSTRAINT "COMMENT\_FK2" FOREIGN KEY ("PostId")

        REFERENCES public."POST" ("PostId") MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

)

CREATE TABLE IF NOT EXISTS public."FOLLOW"

(

    "Follower" character varying COLLATE pg\_catalog."default" NOT NULL,

    "Following" character varying COLLATE pg\_catalog."default" NOT NULL,

    CONSTRAINT "FOLLOW\_pkey" PRIMARY KEY ("Following", "Follower"),

    CONSTRAINT "FOLLOW\_FK1" FOREIGN KEY ("Follower")

        REFERENCES public."DOMAIN" ("DomainName") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE,

    CONSTRAINT "FOLLOW\_FK2" FOREIGN KEY ("Following")

        REFERENCES public."DOMAIN" ("DomainName") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE

)

CREATE TABLE IF NOT EXISTS public."POST"

(

    "PostId" integer NOT NULL,

    "Content" xml NOT NULL,

    "Time" timestamp without time zone NOT NULL,

    "DomainName" character varying(100) COLLATE pg\_catalog."default" NOT NULL,

    CONSTRAINT "PostId\_Unique" PRIMARY KEY ("PostId"),

    CONSTRAINT "POST\_FK" FOREIGN KEY ("DomainName")

        REFERENCES public."DOMAIN" ("DomainName") MATCH SIMPLE

        ON UPDATE CASCADE

        ON DELETE CASCADE

        NOT VALID

)

### Update, Delete, Insert:

**For user table:**

INSERT INTO public."USER"(

    "UserId", "Email", "Username", "Password", "Sex", "Fname", "Lname", "Country", "DomainName", "CreationDate", "Bdate")

    VALUES ("35", "sample@gmail.com", "yagiz124","yagiwezer123asdr", "male", "yagiz", "er", "Turkey", "yagizer124d", "12.01.2022", "11.01.2002");

UPDATE public."USER"

    SET "Country"='Turkey'

    WHERE "UserId"=1;

DELETE FROM public."USER"

    WHERE "UserId"=1;

**For skill table:**

INSERT INTO public."SKILL"(

    "SkillName", "SkillType")

    VALUES ('Yazılım Mühendisliği', 'Teknoloji');

UPDATE public."SKILL"

    SET "SkillType"='Yazılım ve Bilgisayar'

    WHERE "SkillName"='Yazılım Mühendisliği';

DELETE FROM public."SKILL"

    WHERE "SkillName"='Yazılım Mühendisliği';

**For apply table :**

INSERT INTO public."APPLY"(

    "UserId", "DomainName", "AdvertName")

    VALUES (2, 'jwhittock1', 'Peak Games');

UPDATE public."APPLY"

    SET "AdvertName"='YouTube'

    WHERE "UserId"=3;

DELETE FROM public."APPLY"

    WHERE "UserId"=3;

### Check constraints:

For ASSIGMENT Table:

* CONSTRAINT "ASSIGMENT\_TIME\_CHECK" CHECK ("StartTime" <= "EndTime")

For COURSE Table:

* CONSTRAINT "COURSE\_TIME\_CHECK" CHECK ("StartTime" <= "EndTime")

For EDUCATION Table:

* CONSTRAINT "EDU\_TIME\_CHECK" CHECK ("StartTime" <= "EndTime")

For JOB\_HISTORY Table:

* CONSTRAINT "JOB\_TIME\_CHECK" CHECK ("StartTime" <= "EndTime")

For LIVE\_SESSION Table:

* CONSTRAINT "LS\_TIME\_CHECK" CHECK ("StartTime" <= "EndTime")

For STUDENT\_OF Table:

* CONSTRAINT "STUDENT\_GRADE\_CHECK" CHECK ("Grade" <=100 and "Grade" >= 0)

For USER Table:

* CONSTRAINT "USER\_SEX\_CHECK" CHECK ("Sex" in ('Female','Male','Agender','Polygender'))

### Assertions:

**User must be student of the course for uploading homework:**

CREATE

TRIGGER ASSERTION\_UPLOAD\_HW

BEFORE INSERT ON UPLOAD\_HOMEWORK

FOR EACH ROW

BEGIN

IF (SELECT STUDENT\_OF.UserId

FROM STUDENT\_OF, COURSE

WHERE NEW.UserId = STUDENT\_OF.UserId and STUDENT\_OF.CourseId = New.CourseId) is null THEN

RAISE EXCEPTION 'Error: USER MUST BE STUDENT OF THE COURSE'

END;

**User can’t follow itself:**

CREATE

TRIGGER ASSERTION\_FOLLOW

BEFORE INSERT ON FOLLOW

FOR EACH ROW

BEGIN

IF(NEW.follower=NEW.following) THEN

RAISE EXCEPTION 'Error: USER CAN NOT FOLLOW ITSELF'

END;

**User can’t like a post twice:**

CREATE

TRIGGER ASSERION\_LIKE

BEFORE INSERT ON LIKE

FOR EACH ROW

BEGIN

IF(SELECT LIKE.DomainName

FROM LIKE,

WHERE NEW.DomainName = LIKE.DomainName and New.PodtId = LIKE.PostId ) is not null THEN

RAISE EXCEPTION 'Error: USER CAN NOT LIKE A POST AGAIN'

END;

### Triggers:

**Users' creation time must be the same as the current time.**

CREATE

TRIGGER `sqlname`.`check\_usercreationtime`

BEFORE INSERT ON `sqlname`.`user`

FOR EACH ROW

BEGIN

SET NEW.creation\_date = CURRENT\_TIMESTAMP;

END

**Job adverts' posted time must be the same as the current time.**

CREATE

TRIGGER `sqlname`.`check\_advertpostedtime`

BEFORE INSERT ON `sqlname`.`JOB\_ADVERT`

FOR EACH ROW

BEGIN

SET NEW.PostedTime = CURRENT\_TIMESTAMP;

END

**Posts' posted time must be the same as the current time.**

CREATE

TRIGGER `sqlname`.`check\_postpostedtime`

BEFORE INSERT ON `sqlname`.`POST`

FOR EACH ROW

BEGIN

SET NEW.Time = CURRENT\_TIMESTAMP;

END

### Select Statements:

**With 1 Table**

**Students’ grades of a course in descending order**

SELECT "UserId", "Grade"

FROM  public."STUDENT\_OF"

WHERE "STUDENT\_OF"."CourseId" = '12'

ORDER BY "STUDENT\_OF"."Grade" ASC;

**Users that applied to an advert**

SELECT "UserId"

FROM public."APPLY"

WHERE "APPLY"."DomainName" = 'jwhittock1' AND 'Mynet' = "APPLY"."AdvertName";

**Domain name of ones’ that liked a post**

SELECT "LIKE"."DomainName"

FROM public."LIKE"

WHERE "LIKE"."PostId" = 6;

**With 2 Tables**

**Names of universities**

SELECT "ORGANIZATION"."OrgName"

FROM public."ORGANIZATION", public."UNIVERSITY"

WHERE "ORGANIZATION"."OrgId" = "UNIVERSITY"."OrgId";

**Courses don’t have any assistant**

SELECT "COURSE"."CourseId","COURSE"."CourseName"

FROM public."COURSE"

WHERE NOT EXISTS(SELECT 0

FROM public."ASSIST"

WHERE "ASSIST"."CourseId" = "COURSE"."CourseId");

**Users belong to an organization**

SELECT "UserId","Fname","Lname"

FROM public."USER"

WHERE EXISTS(SELECT 0

FROM public."BELONG"

WHERE "USER"."UserId" = "BELONG"."UserId");

**Names of admins in the system**

SELECT "UserId","Fname","Lname"

FROM public."USER"

WHERE EXISTS(SELECT 0

FROM public."BELONG"

WHERE "USER"."UserId" = "BELONG"."UserId" AND "BELONG"."AdministerRight"= '1');

**With 3 Tables**

**Companies who give courses**

SELECT "ORGANIZATION"."OrgName"

FROM public."ORGANIZATION", public."COMPANY"

WHERE "ORGANIZATION"."OrgId" = "COMPANY"."OrgId" AND EXISTS(SELECT 0

FROM public."BELONG"

WHERE "ORGANIZATION"."OrgId" = "BELONG"."OrgId");

**Users with job experience**

 SELECT "USER"."UserId"  
FROM public."USER", public."CV"  
WHERE "CV"."UserId" = "USER"."UserId" AND EXISTS(SELECT 0  
FROM public."JOB\_HISTORY"  
WHERE "JOB\_HISTORY"."CvId" = "CV"."CvId");

**Assistants grouped by courses**

SELECT "USER"."UserId","USER"."Fname","USER"."Lname", "COURSE"."CourseName"  
FROM public."USER", public."ASSIST", public."COURSE"  
WHERE "ASSIST"."UserId" = "USER"."UserId" AND "COURSE"."CourseId" = "ASSIST"."CourseId"  
GROUP BY "COURSE"."CourseName";

### Original Select Statement

**Select statement that gives the users who are in the same city with the organization which they applied their job advert:**

SELECT DISTINCT "Fname", "Lname", "Sex"

FROM public."USER", public."ORGANIZATION", public."APPLY"

WHERE "APPLY"."UserId" = "USER"."UserId" AND "USER"."Country" = "ORGANIZATION"."Country"

ORDER BY "Fname", "Lname" ASC;

**Select statement that gives the all instructors and asssistans:**

SELECT DISTINC "USER"."Fname", "USER"."Lname", "COURSE"."CourseName"

from public."COURSE", public."USER"

where "COURSE"."UserId" = "USER"."UserId"

UNION ALL

select "Fname", "Lname", "CourseName"

from "ASSIST", "USER", "COURSE"

where "ASSIST"."UserId" = "USER"."UserId";

**Select statement that gives the users by grouping who have more than 3 certificate:**

SELECT \*

FROM public."USER"

WHERE "UserId" IN

(SELECT "USER"."UserId"

FROM public."CV", public."USER", public."CERTIFICATE", public."ADD\_CERT"

WHERE "USER"."UserId" = "CV"."UserId" and "CV"."CvId" = "ADD\_CERT"."CvId"

GROUP BY "USER"."UserId"

HAVING COUNT(\*) > 3);

**Select statement that gives the users with their letter grade by looking their grade:**

SELECT "USER"."Fname","USER"."Lname","STUDENT\_OF"."Grade",

case

when "Grade" between 0 and 49 then 'FF'

when "Grade" between 50 and 54 then 'FF'

when "Grade" between 50 and 54 then 'FD'

when "Grade" between 55 and 59 then 'DD'

when "Grade" between 60 and 64 then 'DC'

when "Grade" between 65 and 74 then 'CC'

when "Grade" between 75 and 79 then 'CB'

when "Grade" between 80 and 84 then 'BB'

when "Grade" between 85 and 89 then 'BA'

when "Grade" between 90 and 100 then 'AA'

end as GRADE\_REPORT

from public."STUDENT\_OF" , public."COURSE", public."USER"

Where "STUDENT\_OF"."CourseId" = "COURSE"."CourseId" and "COURSE"."CourseName" ='Engineering' and "USER"."UserId" = "STUDENT\_OF"."UserId";