Contents

ANALYSIS

A Brief Explanation of LinkedIn

LinkedIn is a social platform where users communicate with their environment in business life. It is intended for entrepreneurs, employees, recruiters, internship and job seekers from all possible areas. LinkedIn allows its users to become known and position themselves as an expert. Individual users can create posts on their profiles, send a message to another user, connect with other users, and apply for published job postings. Communities such as companies and groups can create promotional profiles, share posts, send messages and post jobs on the platform to promote themselves.

A Brief Explanation of Moodle

Moodle is an online environment created for educational institutions to create a page of the courses offered and to track and submit homework for registered students. The name "Moodle" is a made-up word and means modular object-oriented online learning environment. It provides an online schooling/teaching enviroment for those who need(because it’s an open-source software).

Aim of LinkedIn

The main purpose of LinkedIn is to connect people in business life(and those who are trying to find a job/worker) and to create a network which built on people and using that to interact.

Aim of Moodle

The main purpose of Moodle is to support the education of the students and provide teachers and students an online schooling/teaching environment.

Main Entities of LinkedIn

User, Group, Company, Notification, Post, Message.

Main Entities of Moodle

Student, Teacher, Grade, School, University, College, Lesson, File, News, Message.

Characteristics of Each Entity in LinkedIn

User: User is the most important entity. Users must have a unique ID, phone number, email, password, birthday, first and last name, position, location, headline schoolID and start date for registration steps. User use email and password informations to sign in to LinkedIn. First name and last name is make user more recognizable. Other attributes like birthday, address and position are using to introducing. User can enter his/her skills, certificates, licenses and past work experiences in profile. These are multivalued features, so user can append more information later.

Group: Group is very similar to facebook pages. Group has a group name which is showing in the group’s profile page. The groups may have numerous members.

Company: As in real life, every company has a name. Company size depends on company. Headline of companies are hit sentences and they change with every product. Companies are working in various industries and they can have websites about themselves.

Notification: Notifications give information about latest activities. They have different types and contents such as connect requests, message notifications, some post notifications etc.

Post: Users or companies can create posts which may have comments, likes and media. Media entity has type attribute which tells if it's a photo(jpeg, jpg, png…) or video(mp4, avi, mkv…). Users/companies also can write comments on posts.

Characteristics of Each Entity in Moodle

Student: Students derive from user entity, so they have user entity's attributes as well. They also have grades, grades must belong to a lesson so they have a lessonId attribute which represents lesson class' lessonId primary key. Also, they all have a gpa to represent their success level on numbers. Students can enroll to lessons and can see the files in those lessons. They can also view the news that attached to their lessons.

Teacher: Teachers also derive from user entity and have its attributes too. Each teacher has a teacher rank like professor or instructor. Teachers can create lessons for students. Then they can upload files to those lessons or edit them. And also share news on lessons to notify students. Lastly, teachers give students their grades.

Grade: Grades are given by teachers to students. They have their id and their lesson's id. Percent attribute holds the importance of a grade for a lesson. For example, midterm grades are generally less important than final grades. Let's say a midterm has 40 percent importance for a lesson, than we can say the final grade has 60 percent and is more important than midterm grade.

School: Schools can be a college or a university. There are a couple of differences between a university and a college. They primarily differ in program offerings and degree types. Universities are larger institutions offering both undergraduate and graduate programs. And college types are community colleges, technical schools and liberal arts colleges. Schools have a name, location and a unique id. Users can register to schools.

University: Universities derive from school entity. They have school's attributes and a type attribute to hold university types.

College: Colleges also derive from school entity and they also have school's attributes. College type attribute indicates a college's type(community colleges, technical schools or liberal arts colleges).

Lesson: Lessons are created by teachers and can be enrolled by students. Teachers can also upload files to a lesson and share news about lessons. Lessons have a name, semester information and a unique id.

File: Files belong to lessons. Teachers can upload and edit them. Students can see(access) files. Files have size, name, date and unique id attributes. File date indicates the upload time from teacher.

News: News also belong to lessons. Teachers can share news and students can view those news. News have date, content and a unique id attribute.

Message: Users can send messages. Message content, date and receiver id are kept in this entity. There can’t be a message if there aren’t any users, therefore message is a weak entity and its strong entity is user entity.

Relationships Among the Entities

One to many relationships:

- “write”: User can write comments.
- ”click”: User can click to post like.
- ”can create”: User, group and company can create posts.
- ”can create”: User can create groups.
- “has”: Post might has comments, post likes and media.
- ”has”: Student have grades.
- “has”: Lesson might have files and news.
- “give”: Teacher can give grades.
- “share”: Teacher can share news.
- ”can see”: Student can see files.
- “upload”: Teacher can upload files.
- “edit”: Teacher can edit files.
- “register”: Users can register a school.
- “send”: User can send messages.
- “can offer”: Company can offer jobs.
- “have”: User and company might have notifications.

Many to many relationships:

- “view user”: Users can view each other.
- “view”: Users can view posts.
- “join”: Users can join groups.
- “can view”: Students can view news.
- “enroll”: Students can enroll to lessons.
- “create”: Teachers can create lessons.
- “message a”: Users and companies can send messages to each other.
- “connect with”: Users and companies can connect with each other.
- “works at”: Users can work at jobs.
- “follow a”: Users can follow companies.
- “view a”: Users and companies can view each other.
- “message user”: Users can send messages to each other.
- “connect”: Users can connect with each other.
- “follow user”: Users can follow each other.

Constraints

Each entity has to has a unique id(except post like).

For every tuple in User entity, phone number, email, password, birthday, first name and last name cannot be null. ‘userId’ should be the primary key for User entity.

Attributes of Group entity name, member count and create date cannot be null. ‘groupId’ is the primary key, ‘creatorUserId’ is a foreign key which references the User entity’s ‘userId’ primary key.

For company; name, type, headline and industry attributes cannot be null. ‘companyId’ is the primary key.

Type, receiver id, content and date attributes of Notification entity cannot be null and ‘notificationId’ is the primary key.

Student and Teacher are subclasses of User entity. ‘userId’ is the primary key for both entities.

School is the superclass of University and College entities. location and name fields cannot be null and ‘schoolId’ is the primary key for all three entities.

‘gradeId’ is the primary key for Grade entity. Grade entity is a weak entity and its strong entity is Student. Lesson id and note percent fields cannot be null. ‘lessonId’ is a foreign key that references the Lesson entity.

Lesson name and semester attributes cannot be null in Lesson entity and ‘lessonId’ is the primary key.

File entity is a weak entity and its strong entity is Lesson. In file entity, file size, file name and date values cannot be null. ‘fileId’ is the primary key.

News entity is a weak entity and its strong entity is Lesson. ‘newsId’ is the primary key for News entity. Content and date fields cannot be null.

‘postId’ is the primary key for Post entity. Post owner’s id and post create date attributes cannot be null.

Post like, media and comment entities are weak entities and their strong entity is the Post entity.

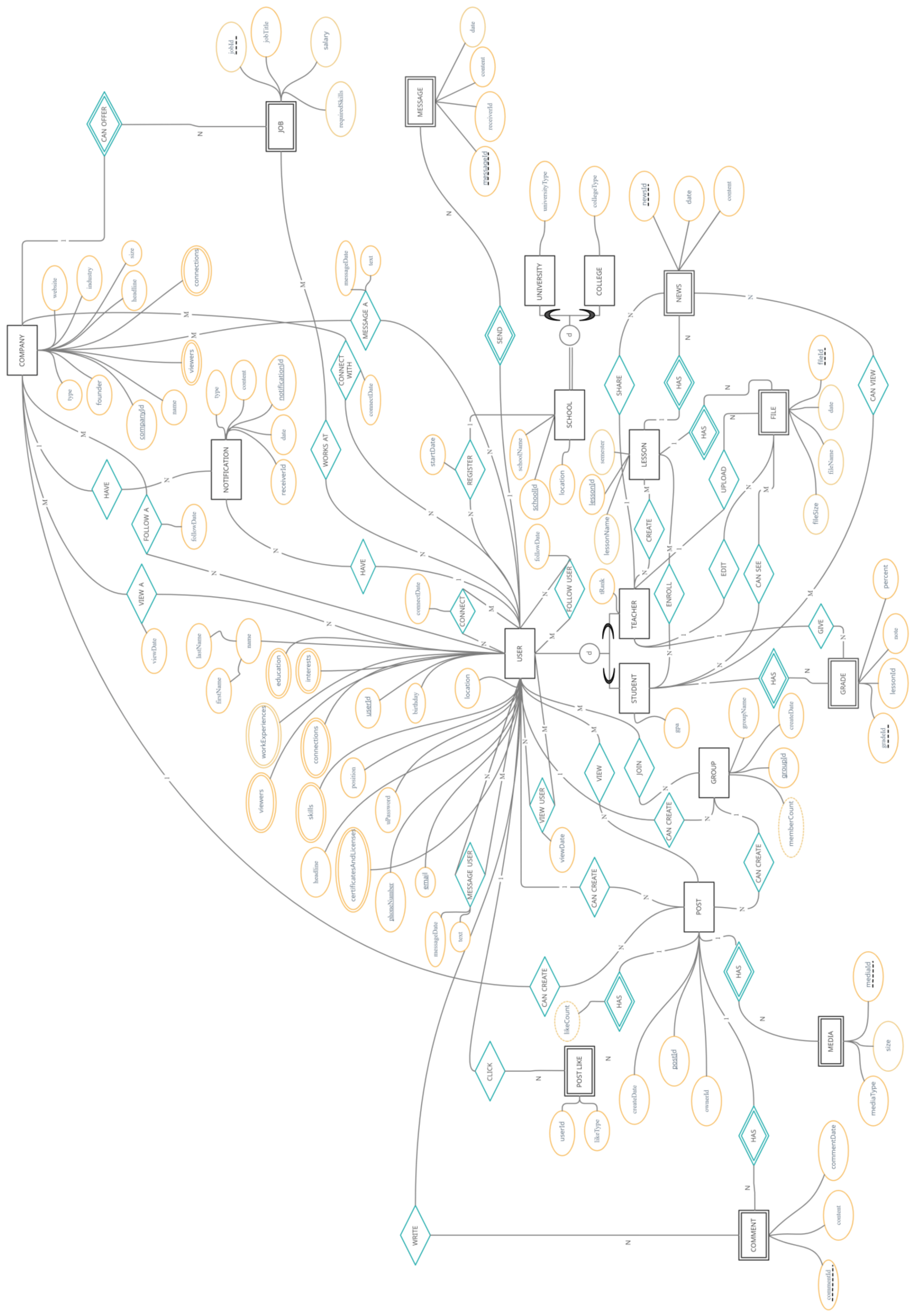
Comment’s content and date attributes cannot be null. ‘commentId’ is the primary key.

Media’s type and size attributes cannot be null. ‘mediaId’ is the primary key.

Message is a weak entity and its strong entity is the user entity. ‘messageId’ is the primary key. Message content and date values cannot be null.

Job is a weak entity and its strong entity is the company entity. ‘jobId’ is the primary key. Job title, salary and required skills attributes cannot be null.

DESIGN-CONCEPTUAL DESIGN



DESIGN-LOGICAL MODEL

Step 1) Mapping of Regular Entity Types

POST(postId, createDate, ownerId)

postId	createDate	ownerId	likeCount

GROUP(groupId, groupName, memberCount, createDate)

groupId	groupName	memberCount	createDate

LESSON(lessonId, lessonName, semester)

lessonId	lessonName	semester

NOTIFICATION(notificationId, type, content, date, receiverId)

notificationId	type	content	date	receiverId

COMPANY(companyId, name, type, size, headline, industry, website)

companyId	name	type	size	headline	industry	website

Step 2) Mapping of Weak Entity Types

POST_LIKE(*postId*, userId, likeType, likeCount)

postId	userId

COMMENT(*postId*, commentId, content, commentDate)

postId	commentId	content	commentDate

MEDIA(*postId*, mediaId, mediaType, size)

postId	mediaId	mediaType	size

GRADE(*userId*, gradeId, *lessonId*, note, percent)

userId	gradeId	lessonId	note	percent

FILE(*lessonId*, fileId, fileSize, fileName, date)

lessonId	fileId	fileSize	fileName	date

NEWS(*lessonId*, newsId, date, content)

lessonId	newsId	date	content

MESSAGE(*userId*, messageId, receiverId, content, date)

userId	messageId	receiverId	content	date

JOB(*companyId*, jobId, jobTitle, salary, requiredSkills)

companyId	jobId	jobTitle	salary	requiredSkills

Step 3) Mapping of Binary 1:1 Relation Types

There aren’t any binary relationships in our EER diagram.

Step 4) Mapping of Binary 1:N Relation Types

COMMENT(*postId*, commentId, content, commentDate, *wUserId*) //from WRITE relationship

postId	commentId	content	commentDate	wUserId

POST LIKE(*postId*, userId, likeType, likeCount, c*UserId*) //from CLICK relationship

postId	userId	likeType	cUserId

POST(postId, createDate, ownerId, *ccCompanyId*) //from CAN CREATE relationship

postId	createDate	ownerId	ccCompanyId

POST(postId, createDate, ownerId, *ccCompanyId, ccUserId*) //from CAN CREATE relationship

postId	createDate	ownerId	ccCompanyId	ccUserId

POST(postId, createDate, ownerId, *ccCompanyId, ccUserId, ccGroupId*) //from CAN CREATE relationship

postId	createDate	ownerId	ccCompanyId	ccUserId	ccGroupId

GROUP(groupId, groupName, memberCount, createDate, *creatorUserId*) //from CAN CREATE relationship

groupId	groupName	memberCount	createDate	creatorUserId

GRADE(*userId*, gradeId, *lessonId*, note, percent, *gTeacherUserId*) //from GIVE relationship

userId	gradeId	lessonId	note	percent	gTeacherUserId

FILE(*lessonId*, fileId, fileSize, fileName, date, *uTeacherUserId*) //from UPLOAD relationship

lessonId	fileId	fileSize	fileName	date	uTeacherUserId

FILE(*lessonId*, fileId, fileSize, fileName, date, *uTeacherUserId, eTeacherUserId*) //from EDIT relationship

lessonId	fileId	fileSize	fileName	date	uTeacherUserId	eTeacherUserId

NEWS(*lessonId*, newsId, date, content, *sTeacherUserId*) //from SHARE relationship

lessonId	newsId	date	content	sTeacherUserId

NOTIFICATION(notificationId, type, content, date, receiverId, *hUserId*) //from HAVE relationship

notificationId	type	content	date	receiverId	hUserId

NOTIFICATION(notificationId, type, content, date, receiverId, *hUserId, hCompanyId*) //from HAVE relationship

notificationId	type	content	date	receiverId	hUserId	hCompanyId

Step 5) Mapping of Binary M:N Relation Types

VIEW USER(*viewerUserId, viewingUserId*, viewDate)

viewerUserId	viewingUserId	viewDate

CONNECT(*connectSenderUserId, connectReceiverUserId*, connectDate)

connectSenderUserId	connectReceiverUserId	connectDate

VIEW(*postId, userId*)

postId	userId

JOIN(*groupId, userId*)

groupId	userId

CONNECT WITH(*connectSenderUserId, connectReceiverCompanyId*, connectDate)

connectSenderUserId	connectReceiverCompanyId	connectDate

FOLLOW USER(*followingUserId, followerUserId*, followDate)

followingUserId	followerUserId	followDate

CREATE(*lessonId, userId*)

lessonId	userId

ENROLL(*lessonId, userId*)

lessonId	userId

CAN SEE(*lessonId, fileId, userId*)

lessonId	fileId	userId

CAN VIEW(*lessonId, newsId, userId*)

lessonId	newsId	userId

WORKS AT(*companyId, jobId, userId*)

companyId	jobId	userId

VIEW A(*companyId, userId*, viewDate)

companyId	userId	viewDate

MESSAGE (*messageSenderUserId, messageReceiverUserId,* text, messageDate)

messageSenderUserId	messageReceiverUserId	text	messageDate

FOLLOW A(*companyId, userId*)

companyId	userId

Step 6) Mapping of Multivalued Attributes

USER CERTIFICATES AND LICENSES(*userId*, uCertificatesAndLicenses)

userId	uCertificatesAndLicenses

USER SKILLS(*userId*, uSkills)

userId	uSkills

USER CONNECTIONS(*userId*, uConnections)

userId	uConnections

USER VIEWERS(*userId*, uViewers)

userId	uViewers

USER WORK EXPERIENCES(*userId*, uWorkExperiences)

userId	uWorkExperiences

USER INTERESTS(*userId*, uInterests)

userId	uInterests

USER EDUCATION(*userId*, uEducation)

userId	uEducation

COMPANY VIEWERS(*companyId*, cViewers)

companyId	cViewers

COMPANY CONNECTIONS(*companyId*, cConnections)

companyId	cConnections

Step 7) Mapping of N-ary Relationship Types

There aren't any n-ary relationships in our EER diagram.

Step 8) Mapping of Specializations or Generalizations

We chose 8A method to convert the "USER-STUDENT/TEACHER" part of our EER diagram because a user can be a student, teacher or a normal user(who has nothing to do with a school, so he/she is just a regular person). And we wanted to create separate tables for STUDENT and TEACHER because we often used some attributes such as "studentId" or "teacherId", so we thought that if we create tables for them, it would make more sense to anyone who examines or uses our database model.

Using 8A: Multiple relations-Superclass and subclasses

USER(userId, phoneNumber, email, uPassword, birthday, firstName, lastName, position, location, headline)

userId	phoneNumber	email	uPassword	birthday	firstName	lastName	position	location	headline

STUDENT(*userId*, gpa)

userId	gpa

TEACHER(*userId*, tRank)

userId	tRank

USER(userId, phoneNumber, email, uPassword, birthday, firstName, lastName, position, location, headline, *schoolId*, startDate) //from REGISTER relationship

userId	phoneNumber	email	uPassword	birthday	firstName	lastName	posit ion	location	headline	schoolId	startDate

For our "SCHOOL-UNIVERSITY/COLLEGE" part of our EER diagram, we chose 8A approach again. Normally, 8B option seems more suitable for disjoint and total relations but we have a 1 to many relationship(register) and it requires us to add "school" table's primary key to "user" table as a foreign key. In this case, we need "school" table because "user" table has a foreign key from it and we need to address it. In more complex schemas, adding an extra table might be a problem but in our schema, "school" table only has 2 subclasses(university and college) so it's not a big problem.

Using 8A: Multiple relations-Superclass and subclasses

SCHOOL(schoolId, location, schoolName)

schoolId	location	schoolName

UNIVERSITY(*schoolId*, universityType)

schoolId	universityType

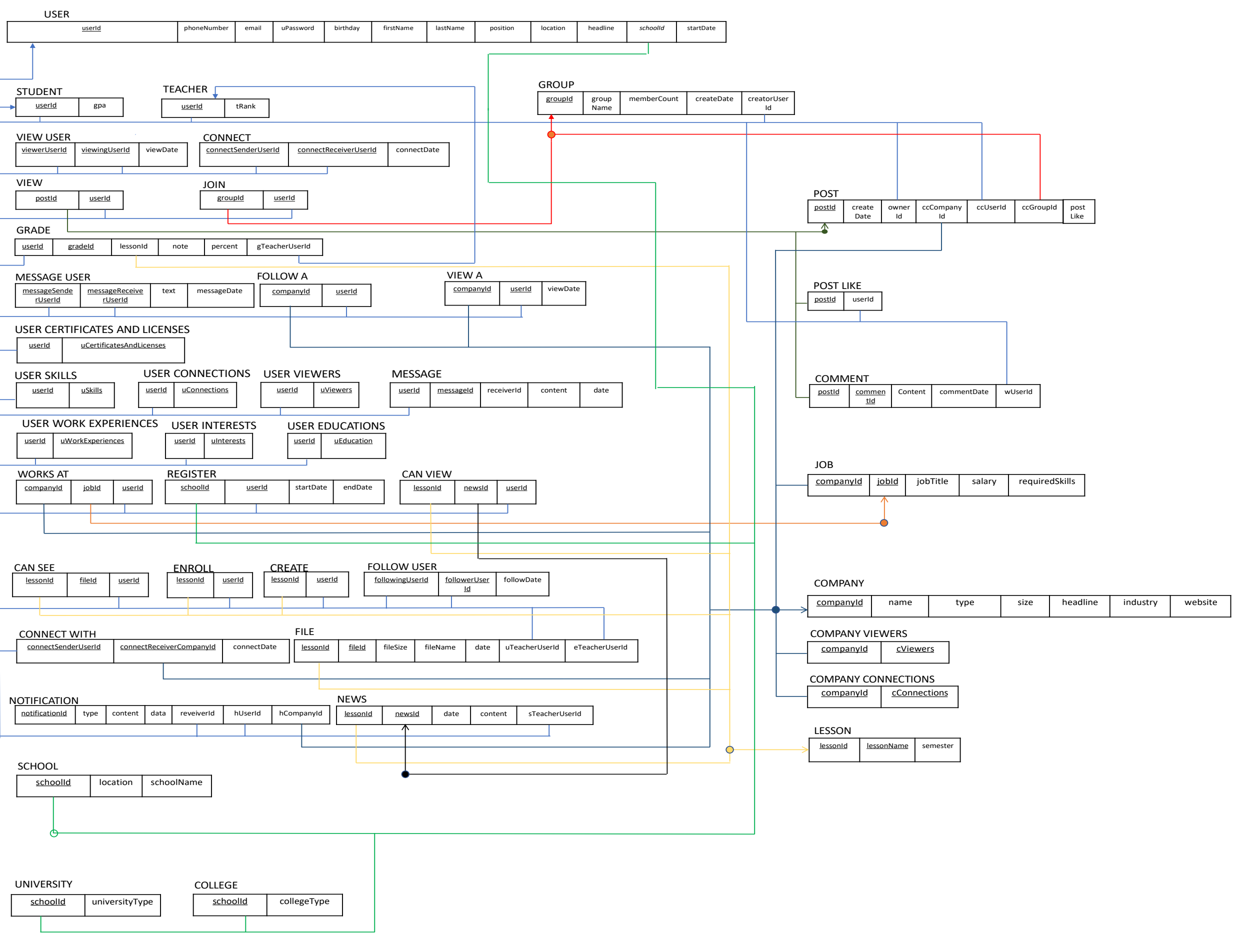
COLLEGE(*schoolId*, collegeType)

<u>schoolId</u>	collegeType

Step 9) Mapping of Union Types (Categories)

There aren’t any union type relations in our EER diagram.

<u>OUR FINAL RELATIONAL DATABASE SCHEMA:</u>



IMPLEMENTATION

CREATE TABLE Statements

```
CREATE TABLE `can_see` (
  `lessonId` INT NOT NULL,
  `fileId` INT NOT NULL,
  `userId` INT NOT NULL,
  PRIMARY KEY (`lessonId`,`fileId`,`userId`),
  KEY `CANSEEUSERFK` (`userId`),
  CONSTRAINT `CANSEELESSONFK` FOREIGN KEY (`fileId`) REFERENCES `fil
e` (`fileId`) ON DELETE CASCADE ON UPDATE CASCADE,
  FOREIGN KEY (`lessonId`) REFERENCES `lesson` (`lessonId`) ON DELET
E CASCADE ON UPDATE CASCADE,
  CONSTRAINT `CANSEEUSERFK` FOREIGN KEY (`userId`) REFERENCES `user`
 (`userId`) ON DELETE CASCADE ON UPDATE CASCADE
)

CREATE TABLE `can_view` (
  `lessonId` INT NOT NULL,
  `newsId` INT NOT NULL,
  `userId` INT NOT NULL,
  PRIMARY KEY (`lessonId`,`newsId`,`userId`),
  KEY `CANVIEWUSERFK` (`userId`),
  CONSTRAINT `CANVIEWLESSONFK` FOREIGN KEY (`lessonId`, `newsId`) RE
FERENCES `news` (`lessonId`, `newsId`) ON DELETE CASCADE ON UPDATE C
ASCADE,
  CONSTRAINT `CANVIEWUSERFK` FOREIGN KEY (`userId`) REFERENCES `user
` (`userId`) ON DELETE CASCADE ON UPDATE CASCADE
)

CREATE TABLE college (
  schoolId INT auto_increment NOT NULL,
  collegeType varchar(40) NOT NULL,
  PRIMARY KEY (schoolId),
  CONSTRAINT COLSCHOOLFK
  FOREIGN KEY (schoolId) REFERENCES school (schoolId)
  ON DELETE CASCADE
  ON UPDATE CASCADE
)

CREATE TABLE `comment` (
  `postId` INT NOT NULL,
  `commentId` INT NOT NULL,
  `content` varchar(40) NOT NULL,
  `commentDate` date NOT NULL,
  `wUserId` INT DEFAULT NULL,
  PRIMARY KEY (`postId`,`commentId`),
  KEY `COMMENTUSERFK` (`wUserId`),
  CONSTRAINT `COMMENTUSERFK` FOREIGN KEY (`wUserId`) REFERENCES `use
r` (`userId`) ON DELETE SET NULL ON UPDATE CASCADE
)

CREATE TABLE company (
  companyId INT auto_increment NOT NULL,
  name varchar(30) NOT NULL,
  type varchar(15) NOT NULL,
  size varchar(15) DEFAULT NULL,
  headline varchar(50) NOT NULL,
  industry varchar(20) NOT NULL,
  website varchar(35) DEFAULT NULL,
  PRIMARY KEY (companyId)
)

CREATE TABLE companyConnections (
  companyId INT NOT NULL,
  cConnections varchar(50) NOT NULL,
  PRIMARY KEY (companyId,cConnections),
  CONSTRAINT CONCOMPFK FOREIGN KEY (companyId) REFERENCES company (c
ompanyId) ON DELETE CASCADE ON UPDATE CASCADE
)

CREATE TABLE companyViewers (
  companyId INT NOT NULL,
  cViewers varchar(50) NOT NULL,
  PRIMARY KEY (companyId,cViewers),
  CONSTRAINT VIEWCOMPFK FOREIGN KEY (companyId) REFERENCES company (
companyId) ON DELETE CASCADE ON UPDATE CASCADE
)

CREATE TABLE `connect` (
  `connectSenderUserId` INT NOT NULL,
  `connectReceiverUserId` INT NOT NULL,
  `connectDate` date NOT NULL,
  PRIMARY KEY (`connectSenderUserId`,`connectReceiverUserId`),
  KEY `CRECEIVERUSERFK` (`connectReceiverUserId`),
  CONSTRAINT `CRECEIVERUSERFK` FOREIGN KEY (`connectReceiverUserId`)
 REFERENCES `user` (`userId`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `CSENDERUSERFK` FOREIGN KEY (`connectSenderUserId`) REF
ERENCES `user` (`userId`) ON DELETE CASCADE ON UPDATE CASCADE
)

CREATE TABLE `connect_with` (
  `connectSenderUserId` INT NOT NULL,
  `connectReceiverCompanyId` INT NOT NULL,
  `connectDate` date NOT NULL,
  PRIMARY KEY (`connectSenderUserId`,`connectReceiverCompanyId`),
  KEY `CWRECEIVERUSERFK` (`connectReceiverCompanyId`),
  CONSTRAINT `CWRECEIVERUSERFK` FOREIGN KEY (`connectReceiverCompany
Id`) REFERENCES `company` (`companyId`) ON DELETE CASCADE ON UPDATE 
CASCADE,
  CONSTRAINT `CWSENDERUSERFK` FOREIGN KEY (`connectSenderUserId`) RE
FERENCES `user` (`userId`) ON DELETE CASCADE ON UPDATE CASCADE
)

CREATE TABLE `create` (
  `lessonId` INT NOT NULL,
  `userId` INT NOT NULL,
  PRIMARY KEY (`lessonId`,`userId`),
  KEY `CREATEUSERFK` (`userId`),
  CONSTRAINT `CREATELESSONFK` FOREIGN KEY (`lessonId`) REFERENCES `l
esson` (`lessonId`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `CREATEUSERFK` FOREIGN KEY (`userId`) REFERENCES `user`
 (`userId`) ON DELETE CASCADE ON UPDATE CASCADE
)

CREATE TABLE `enroll` (
  `lessonId` INT NOT NULL,
  `userId` INT NOT NULL,
  PRIMARY KEY (`lessonId`,`userId`),
  KEY `ENROLLUSERFK` (`userId`),
  CONSTRAINT `ENROLLLESSONFK` FOREIGN KEY (`lessonId`) REFERENCES `l
esson` (`lessonId`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `ENROLLUSERFK` FOREIGN KEY (`userId`) REFERENCES `user`
 (`userId`) ON DELETE CASCADE ON UPDATE CASCADE
)
```

```
CREATE TABLE `file` (
  `lessonId` INT NOT NULL,
  `fileId` INT AUTO_INCREMENT NOT NULL,
  `filesize` varchar(40) NOT NULL,
  `fileName` varchar(50) NOT NULL,
  `date` date NOT NULL,
  `uTeacherUserId` INT DEFAULT NULL,
  `eTeacherUserId` INT DEFAULT NULL,
  PRIMARY KEY (`fileId`),
  KEY `FILEUPFK` (`uTeacherUserId`),
  KEY `FILEEDFK` (`eTeacherUserId`),
  CONSTRAINT `FILEEDFK` FOREIGN KEY (`eTeacherUserId`) REFERENCES `t
eacher` (`userId`) ON DELETE SET NULL ON UPDATE CASCADE,
  CONSTRAINT `FILEUPFK` FOREIGN KEY (`uTeacherUserId`) REFERENCES `t
eacher` (`userId`) ON DELETE SET NULL ON UPDATE CASCADE
)
CREATE TABLE `follow_a` (
  `companyId` INT NOT NULL,
  `userId` INT NOT NULL,
  `followDate` date NOT NULL,
  PRIMARY KEY (`companyId`,`userId`),
  KEY `FOLLOWUSERFK` (`userId`),
  CONSTRAINT `FOLLOWCOMPFK` FOREIGN KEY (`companyId`) REFERENCES `co
mpany` (`companyId`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `FOLLOWUSERFK` FOREIGN KEY (`userId`) REFERENCES `user`
 (`userId`) ON DELETE CASCADE ON UPDATE CASCADE
)

CREATE TABLE `follow_user` (
  `followingUserId` INT NOT NULL,
  `followerUserId` INT NOT NULL,
  `followDate` date NOT NULL,
  PRIMARY KEY (`followingUserId`,`followerUserId`),
  KEY `FOLLOWERUSERFK` (`followerUserId`),
  CONSTRAINT `FOLLOWERUSERFK` FOREIGN KEY (`followerUserId`) REFEREN
CES `user` (`userId`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `FOLLOWINGUSERFK` FOREIGN KEY (`followingUserId`) REFER
ENCES `user` (`userId`) ON DELETE CASCADE ON UPDATE CASCADE
)

CREATE TABLE `grade` (
  `userId` INT NOT NULL,
  `gradeId` INT NOT NULL,
  `lessonId` INT NOT NULL,
  `note` int NOT NULL,
  `percent` int NOT NULL,
  `gTeacherUserId` INT DEFAULT NULL,
  PRIMARY KEY (`userId`,`gradeId`),
  KEY `GRADETEACHERFK` (`gTeacherUserId`),
  KEY `GRADELESSONFK_idx` (`lessonId`),
  CONSTRAINT `GRADELESSONFK` FOREIGN KEY (`lessonId`) REFERENCES `le
sson` (`lessonId`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `GRADESTUDENTFK` FOREIGN KEY (`userId`) REFERENCES `stu
dent` (`userId`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `GRADETEACHERFK` FOREIGN KEY (`gTeacherUserId`) REFEREN
CES `teacher` (`userId`) ON DELETE SET NULL ON UPDATE CASCADE,
  CONSTRAINT `grade_chk_1` CHECK (((`note` >= 0) and (`note` <= 100)
)),
  CONSTRAINT `grade_chk_2` CHECK (((`percent` >= 0) and (`percent` <
= 50)))
)

CREATE TABLE `group`(
  groupId INT auto_increment NOT NULL,
  groupName varchar(30) NOT NULL,
  memberCount int NOT NULL,
  createDate date NOT NULL,
  creatorUserId INT DEFAULT NULL,
  PRIMARY KEY (groupId),
  KEY GROUPUSERFK (creatorUserId),
  CONSTRAINT GROUPUSERFK FOREIGN KEY (creatorUserId) REFERENCES user
 (userId) ON DELETE SET NULL ON UPDATE CASCADE
)

CREATE TABLE job (
  companyId INT NOT NULL,
  jobId INT NOT NULL AUTO_INCREMENT,
  jobTitle varchar(30) NOT NULL,
  salary int NOT NULL,
  requiredSkills varchar(100) NOT NULL,
  PRIMARY KEY (jobId),
  CONSTRAINT JOBCOMPFK FOREIGN KEY (companyId) REFERENCES company (c
ompanyId) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT job_chk_1 CHECK ((salary > 0))
)

CREATE TABLE `join` (
  `groupId` INT NOT NULL,
  `userId` INT NOT NULL,
  PRIMARY KEY (`groupId`,`userId`),
  KEY `JOINUSERFK` (`userId`),
  CONSTRAINT `JOINGROUPFK` FOREIGN KEY (`groupId`) REFERENCES `group
` (`groupId`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `JOINUSERFK` FOREIGN KEY (`userId`) REFERENCES `user` (
`userId`) ON DELETE CASCADE ON UPDATE CASCADE
)

CREATE TABLE lesson (
  lessonId INT auto_increment NOT NULL,
  lessonName varchar(40) NOT NULL,
  semester varchar(15) NOT NULL,
  PRIMARY KEY (lessonId)
)

CREATE TABLE `message` (
  `userId` INT NOT NULL,
  `messageId` INT AUTO_INCREMENT NOT NULL,
  `receiverId` INT NOT NULL,
  `content` varchar(100) NOT NULL,
  `date` date NOT NULL,
  PRIMARY KEY (`messageId`),
  CONSTRAINT `MESSAGEUSERFK` FOREIGN KEY (`userId`) REFERENCES `user
` (`userId`) ON DELETE CASCADE ON UPDATE CASCADE
)

CREATE TABLE `news` (
  `lessonId` INT NOT NULL,
  `newsId` INT NOT NULL,
  `content` varchar(100) NOT NULL,
  `date` date NOT NULL,
  `sTeacherUserId` INT DEFAULT NULL,
  PRIMARY KEY (`lessonId`,`newsId`),
  KEY `NEWSTEACHERFK` (`sTeacherUserId`),
  CONSTRAINT `NEWSTEACHERFK` FOREIGN KEY (`sTeacherUserId`) REFERENC
ES `teacher` (`userId`) ON DELETE SET NULL ON UPDATE CASCADE
```

```
)

CREATE TABLE `notification` (
  `notificationId` INT auto_increment NOT NULL,
  `type` varchar(15) NOT NULL,
  `receiverId` INT NOT NULL,
  `content` varchar(50) NOT NULL,
  `date` date NOT NULL,
  `hUserId` INT DEFAULT NULL,
  `hCompanyId` INT DEFAULT NULL,
  PRIMARY KEY (`notificationId`),
  KEY `NOTIFUSERFK` (`hUserId`),
  KEY `NOTIFCOMPFK` (`hCompanyId`),
  CONSTRAINT `NOTIFCOMPFK` FOREIGN KEY (`hCompanyId`) REFERENCES `co
mpany` (`companyId`) ON DELETE SET NULL ON UPDATE CASCADE,
  CONSTRAINT `NOTIFUSERFK` FOREIGN KEY (`hUserId`) REFERENCES `user`
 (`userId`) ON DELETE SET NULL ON UPDATE CASCADE
)

CREATE TABLE `post` (
  `postId` INT auto_increment NOT NULL,
  `createDate` date NOT NULL,
  `ownerId` INT NOT NULL,
  likeCount INT DEFAULT(0),
  `ccCompanyId` INT DEFAULT NULL,
  `ccUserId` INT DEFAULT NULL,
  `ccGroupId` INT DEFAULT NULL,
  content varchar(50) DEFAULT NULL,
  PRIMARY KEY (`postId`),
  KEY `POSTCOMPFK` (`ccCompanyId`),
  KEY `POSTUSERFK` (`ccUserId`),
  KEY `POSTGROUPFK` (`ccGroupId`),
  CONSTRAINT `POSTCOMPFK` FOREIGN KEY (`ccCompanyId`) REFERENCES `co
mpany` (`companyId`) ON DELETE SET NULL ON UPDATE CASCADE,
  CONSTRAINT `POSTGROUPFK` FOREIGN KEY (`ccGroupId`) REFERENCES `gro
up` (`groupId`) ON DELETE SET NULL ON UPDATE CASCADE,
  CONSTRAINT `POSTUSERFK` FOREIGN KEY (`ccUserId`) REFERENCES `user`
 (`userId`) ON DELETE SET NULL ON UPDATE CASCADE
)

CREATE TABLE `post like` (
  `postId` INT NOT NULL,
  `userId` INT NOT NULL,
  PRIMARY KEY (`postId`),
  KEY `POSTLIKEUSERFK` (`cUserId`),
  CONSTRAINT `POSTLIKEUSERFK` FOREIGN KEY (`cUserId`) REFERENCES `us
er` (`userId`) ON DELETE SET NULL ON UPDATE CASCADE
)

CREATE TABLE `register` (
  `schoolId` INT NOT NULL,
  `userId` INT NOT NULL,
  PRIMARY KEY (`schoolId`,`userId`),
  FOREIGN KEY (`schoolId`) REFERENCES `school` (`schoolId`) ON DELET
E CASCADE ON UPDATE CASCADE,
  FOREIGN KEY (`userId`) REFERENCES `user` (`userId`) ON DELETE CASC
ADE ON UPDATE CASCADE
)

CREATE TABLE `school` (
  `schoolId` INT NOT NULL auto_increment,
  `location` varchar(30) NOT NULL,
  `schoolName` varchar(40) NOT NULL,
  PRIMARY KEY (`schoolId`)
)

CREATE TABLE `student` (
  `userId` INT NOT NULL,
  `gpa` double DEFAULT 0.0,
  PRIMARY KEY (`userId`),
  CONSTRAINT `STUDENTFK` FOREIGN KEY (`userId`) REFERENCES `user` (`
userId`) ON UPDATE CASCADE
)

CREATE TABLE `teacher` (
  `userId` INT auto_increment NOT NULL,
  `tRank` varchar(20) NOT NULL,
  PRIMARY KEY (`userId`),
  CONSTRAINT `TEACHERFK` FOREIGN KEY (`userId`) REFERENCES `user` (`
userId`) ON UPDATE CASCADE
)

CREATE TABLE university (
  schoolId INT NOT NULL auto_increment,
  universityType varchar(40) NOT NULL,
  PRIMARY KEY (schoolId),
  CONSTRAINT UNISCHOOLFK
  FOREIGN KEY (schoolId) REFERENCES school (schoolId)
  ON DELETE CASCADE
  ON UPDATE CASCADE
)

CREATE TABLE user (
  userId INT auto_increment NOT NULL,
  phoneNumber char(10) NOT NULL,
  email varchar(30) NOT NULL,
  uPassword varchar(20) NOT NULL,
  birthday date NOT NULL,
  firstName varchar(15) NOT NULL,
  lastName varchar(20) NOT NULL,
  position varchar(30) DEFAULT NULL,
  location varchar(30) DEFAULT NULL,
  headline varchar(50) DEFAULT NULL,
  schoolId INT DEFAULT NULL,
  startDate date DEFAULT NULL,
  PRIMARY KEY (userId),
  UNIQUE KEY phoneNumber (phoneNumber),
  UNIQUE KEY email (email),
  KEY schoolId_idx (schoolId),
  CONSTRAINT USERSCHOOLFK
  FOREIGN KEY (schoolId) REFERENCES school (schoolId)
)

CREATE TABLE `user_certificates_and_licenses` (
  `userId` INT NOT NULL,
  `uCertificatesAndLicenses` varchar(50) NOT NULL,
  PRIMARY KEY (`userId`,`uCertificatesAndLicenses`),
  CONSTRAINT `CERTUSERFK` FOREIGN KEY (`userId`) REFERENCES `user` (
`userId`) ON DELETE CASCADE ON UPDATE CASCADE
)

CREATE TABLE `user_connections` (
  `userId` INT NOT NULL,
  `uConnections` varchar(50) NOT NULL,
  PRIMARY KEY (`userId`,`uConnections`),
  CONSTRAINT `CONUSERFK` FOREIGN KEY (`userId`) REFERENCES `user` (`
userId`) ON DELETE CASCADE ON UPDATE CASCADE
)

CREATE TABLE `user_education` (
  `userId` INT NOT NULL,
  `uEducation` varchar(50) NOT NULL,
  PRIMARY KEY (`userId`,`uEducation`),
  CONSTRAINT `EDUCUSERFK` FOREIGN KEY (`userId`) REFERENCES `user` (
`userId`) ON DELETE CASCADE ON UPDATE CASCADE
)

CREATE TABLE `user_interests` (
  `userId` INT NOT NULL,
  `uInterests` varchar(50) NOT NULL,
  PRIMARY KEY (`userId`,`uInterests`),
  CONSTRAINT `INTERESTUSERFK` FOREIGN KEY (`userId`) REFERENCES `use
r` (`userId`) ON DELETE CASCADE ON UPDATE CASCADE
)

CREATE TABLE `user_skills` (
  `userId` INT NOT NULL,
  `uSkills` varchar(50) NOT NULL,
  PRIMARY KEY (`userId`,`uSkills`),
  CONSTRAINT `SKILLUSERFK` FOREIGN KEY (`userId`) REFERENCES `user`
(`userId`) ON DELETE CASCADE ON UPDATE CASCADE
)

CREATE TABLE `user_viewers` (
  `userId` INT NOT NULL,
  `uViewers` varchar(50) NOT NULL,
  PRIMARY KEY (`userId`,`uViewers`),
  CONSTRAINT `USERVIEWERFK` FOREIGN KEY (`userId`) REFERENCES `user`
 (`userId`) ON DELETE CASCADE ON UPDATE CASCADE
)

CREATE TABLE `user_work_experiences` (
  `userId` INT NOT NULL,
  `uWorkExperiences` varchar(50) NOT NULL,
  PRIMARY KEY (`userId`,`uWorkExperiences`),
  CONSTRAINT `EXPUSERFK` FOREIGN KEY (`userId`) REFERENCES `user` (`
userId`) ON DELETE CASCADE ON UPDATE CASCADE
```

```
)

CREATE TABLE `view` (
  `postId` INT NOT NULL,
  `userId` INT NOT NULL,
  PRIMARY KEY (`postId`,`userId`),
  KEY `VIEWUSERFK` (`userId`),
  CONSTRAINT `VIEWPOSTFK` FOREIGN KEY (`postId`) REFERENCES `post` (
`postId`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `VIEWUSERFK` FOREIGN KEY (`userId`) REFERENCES `user` (
`userId`) ON DELETE CASCADE ON UPDATE CASCADE
)

CREATE TABLE `view_a` (
  `companyId` INT NOT NULL,
  `userId` INT NOT NULL,
  `viewDate` date NOT NULL,
  PRIMARY KEY (`companyId`,`userId`),
  KEY `VIEWAUSERFK` (`userId`),
  CONSTRAINT `VIEWACOMPFK` FOREIGN KEY (`companyId`) REFERENCES `com
pany` (`companyId`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `VIEWAUSERFK` FOREIGN KEY (`userId`) REFERENCES `user`
(`userId`) ON DELETE CASCADE ON UPDATE CASCADE
)

CREATE TABLE `view_user` (
  `viewerUserId` INT NOT NULL,
  `viewingUserId` INT NOT NULL,
  `viewDate` date NOT NULL,
  PRIMARY KEY (`viewerUserId`,`viewingUserId`),
  KEY `VIEWINGUSERFK` (`viewingUserId`),
  CONSTRAINT `VIEWERUSERFK` FOREIGN KEY (`viewerUserId`) REFERENCES
`user` (`userId`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `VIEWINGUSERFK` FOREIGN KEY (`viewingUserId`) REFERENCE
S `user` (`userId`) ON DELETE CASCADE ON UPDATE CASCADE
)

CREATE TABLE `works_at` (
  `companyId` INT NOT NULL,
  `jobId` INT NOT NULL,
  `userId` INT NOT NULL,
  PRIMARY KEY (`companyId`,`jobId`,`userId`),
  KEY `WORKUSERFK` (`userId`),
  CONSTRAINT `WORKCOMPFK` FOREIGN KEY (`companyId`, `jobId`) REFEREN
CES `job` (`companyId`, `jobId`) ON DELETE CASCADE ON UPDATE CASCADE
,
  CONSTRAINT `WORKUSERFK` FOREIGN KEY (`userId`) REFERENCES `user` (
`userId`) ON DELETE CASCADE ON UPDATE CASCADE
)
```

INSERT Statements

```
insert into can_see
values
(1,78),
(2,78),
(3,78),
(2,79),
(3,79),
(4,79),
(5,79),
(5,80),
(4,80),
(4,81),
(4,82),
(5,81),
(5,83),
(6,84),
(6,81),
(6,79),
(7,78),
(7,85);

insert into user_certificates_and_licenses
values
(78,'Python from John Hopkins'),
(78,'C++ from Yale'),
(78,'Node.js from Harvard'),
(78,'Git from Google'),
(78,'React.js from Harvard'),
(78,'C++ from Berlin'),
(79,'French from Dusseldorf'),
(79,'German from Dusseldorf'),
(79,'Turkish from Dusseldorf'),
(80,'Figma from Udemy'),
(80,'Vue.js from Udemy'),
(80,'Laravel from Google'),
(81,'Jinja from Utah State'),
(81,'Flask from Google'),
(82,'VR from Udemy'),
(82,'AR from Harvard'),
(83,'File Protocols from Udemy'),
(83,'Linux from Google'),
(84,'MS-DOS from Udemy'),
(85,'Network from Harvard'),
(85,'Network from Udemy');

insert into
    company (name,type,size,headline,industry,website)
values(
        'Tesla',
        'company',
        '349657',
        'Future is Tesla',
        'tech',
        "www.tesla.com"
    ),
(
        'Amazon',
        'company',
        '701269',
        'Buy anything',
        'e-commerce',
        "www.amazon.com"
    ),
(
        'Apple',
        'company',
        '142984',
        'New Macbook',
        'tech',
        "www.apple.com"
    ),
(
        'Shell',
        'company',
        '100000',
        'Pure oil power',
        'oil',
        "www.shell.com"
    );

insert into connect()
values
(78,79,'2012-04-04'),
(78,80,'2012-04-04'),
(78,81,'2012-04-04'),
(78,82,'2012-04-04'),
(78,83,'2012-04-04'),
(78,84,'2012-04-04'),
(78,85,'2012-04-04'),
(81,85,'2012-04-04'),
(82,84,'2012-04-04'),
(81,83,'2012-04-04'),
(82,83,'2012-04-04'),
(84,78,'2012-04-04'),
(84,85,'2012-04-04'),
(85,79,'2012-04-04');

insert into user_education()
values
(78,32),
(79,32),
(80,32),
(80,31),
(81,30),
(82,29),
(83,29),
(84,28),
(85,28);
insert into enroll
values
(1,78),
(2,78),
(3,78),
(4,78),
(4,81),
(3,81),
(6,81),
(5,79),
(6,79),
(7,79),
(8,79),
(9,80),
(10,80);

insert into file(lessonId,filesize,fileName,`date`)
values
(1,'5mb','Midterm Result','2001-05-12'),
(2,'8mb','2. Quiz Result','2001-05-12'),
(2,'4mb','Midterm Result','2001-05-12'),
(3,'2mb','Term Project Result','2001-05-12'),
(4,'9mb','Midterm Result','2001-05-12'),
(5,'1mb','Midterm Result','2001-05-12'),
(6,'2mb','1. Quiz Result','2001-05-12'),
(7,'3mb','Midterm Result','2001-05-12');

insert into follow_a
values
(1,78),
(2,78),
(3,78),
(4,78),
(1,79),
(2,80),
(3,80),
(4,81),
(1,81),
(2,82),
(3,82),
(4,82),
(1,83),
(2,84),
(3,85),
(4,85);

insert into follow_user
values
(78,79,'2000-02'),
(79,80,'2000-02'),
(78,81,'2000-02'),
(78,82,'2000-02'),
(79,82,'2000-02'),
(79,81,'2000-02'),
(80,79,'2000-02'),
(80,78,'2000-02'),
(80,81,'2000-02'),
(80,82,'2000-02'),
(84,82,'2000-02'),
(84,81,'2000-02'),
(84,80,'2000-02'),
(85,78,'2000-02'),
(85,79,'2000-02'),
(85,80,'2000-02'),
(85,82,'2000-02');

insert into
    grade
values(
        '9',
        '165469359',
        '1',
        70,
        40,
        '13'
    ),
(
        '9',
        '165469360',
        '2',
        56,
        40,
        '14'
    ),
(
        '10',
        '165469361',
        '3',
        75,
        30,
        '15'
    ),(
        '11',
        '165469362',
        '4',
        35,
        40,
        '16'
    ),(
        '12',
        '165469363',
        '5',
        80,
        40,
        '16'
    );

insert into `group`(groupName,memberCount,createDate,creatorUserId)
values
('Waltz',1,'1990-12-31',82),
('Programming',1,'1994-12-31',82),
('Martial Arts',1,'1998-12-31',83),
('Theatre',1,'2000-12-31',85),
('Music',1,'2012-12-31',79),
('IEEE',1,'1999-12-31',84);

insert into user_interests
values
(78,'Waltz'),
(79,'RnB'),
(80,'Piano'),
(81,'Guitar'),
(82,'History'),
(82,'Novel'),
(83,'Travelling'),
(84,'Novel'),
(84,'Music'),
(84,'Hike'),
(85,'Music'),
(85,'History Books'),
(80,'Gaming'),
(80,'Music'),
(78,'Badminton'),
(79,'Music'),
(81,'Ping Pong'),
(82,'Ping Pong'),
(83,'Cinema'),
(79,'Nature'),
(81,'Music'),
(80,'Illusturations'),
(80,'Unity'),
(83,'Music');

insert into job(companyId,jobTitle,salary,requiredSkills)
values
(1,'Region Manager',10000,''),
(1,'Data Engineer',12354,''),
(2,'CTO',356123,''),
(2,'Senior Backend Devoloper',14862,''),
(2,'Sales Manager',32684,''),
(3,'Designer',78952,''),
(3,'Market Consultant',12014,''),
(4,'CEO',985621,''),
(4,'Frontend Developer',12132,''),
(4,'Mechanical Engineer',20150,''),
(4,'Store Security',14128,'');

insert into `join`
values
(1,78),
(1,79),
(1,80),
(1,81),
(2,82),
(2,83),
(2,84),
(3,85),
(4,78),
(4,79),
(4,80),
(4,81),
(5,82),
(6,83),
(6,84),
(6,85);
```

```
insert into lesson(lessonName, semester)
values
('Math 101', 'semester 1'),
('Physics 101', 'semester 1'),
('Algorithm and Programming 102', 'semester 2'),
('Introduction to Computer Science', 'semester 2'),
('Data Structures', 'semester 3'),
('Automata Theory', 'semester 3'),
('Differential Equations', 'semester 4'),
('Digital Computer Design', 'semester 4'),
('Programming Languages', 'semester 4'),
('Operatin Systems', 'semester 5'),
('Database Managment Systems', 'semester 5'),
('Intern Program', 'semester 6'),
('Microcontroller', 'semester 6'),
('Algorithm Analysis', 'semester 6'),
('Software Architecture', 'semester 7'),
('Thesis', 'semester 8'),
('Math-2','semester 2'),
('Heat Transfer','semester 4'),
('Economics','semester 3'),
('Illustration','semester 5');

insert into message(userId,receiverId,content,`date`)
values
(78,79,'Hi, do u wanna hangout?','2012-03-05'),
(79,80,'Hi, do u wanna hangout?','2012-03-05'),
(81,82,'Hi, do u wanna hangout?','2012-03-05'),
(81,82,'Hi, do u wanna hangout again?','2012-03-05'),
(83,84,'Hi, do u wanna hangout?','2012-03-05'),
(78,84,'Hi, do u wanna hangout?','2012-03-05'),
(79,81,'Hi, do u wanna hangout?','2012-03-05');

insert into post(createDate,ownerId,ccUserId,content)
values
('2012-08-
21',78,78,'ac tortor dignissim convallis aenean et tortor at risus v
iverra adipiscing at in tellus integer feugiat scelerisque varius mo
rbi enim nunc faucibus a pellentesque'),
('2012-08-
21',80,80,'ac tortor dignissim convallis aenean et tortor at risus v
iverra adipiscing at in tellus integer feugiat scelerisque varius mo
rbi enim nunc faucibus a pellentesque'),
('2012-08-
21',81,81,'ac tortor dignissim convallis aenean et tortor at risus v
iverra adipiscing at in tellus integer feugiat scelerisque varius mo
rbi enim nunc faucibus a pellentesque'),
('2012-08-
21',82,82,'ac tortor dignissim convallis aenean et tortor at risus v
iverra adipiscing at in tellus integer feugiat scelerisque varius mo
rbi enim nunc faucibus a pellentesque'),
('2012-08-
21',83,83,'ac tortor dignissim convallis aenean et tortor at risus v
iverra adipiscing at in tellus integer feugiat scelerisque varius mo
rbi enim nunc faucibus a pellentesque'),
('2012-08-
21',84,84,'ac tortor dignissim convallis aenean et tortor at risus v
iverra adipiscing at in tellus integer feugiat scelerisque varius mo
rbi enim nunc faucibus a pellentesque'),
('2012-08-
21',85,85,'ac tortor dignissim convallis aenean et tortor at risus v
iverra adipiscing at in tellus integer feugiat scelerisque varius mo
rbi enim nunc faucibus a pellentesque');

insert into post(createDate,ownerId,ccCompanyId,content)
values
('2017-04-
02',1,1,'vitae elementum curabitur vitae nunc sed velit dignissim'),
('2017-04-
02',2,2,'vitae elementum curabitur vitae nunc sed velit dignissim'),
('2017-04-
02',3,3,'vitae elementum curabitur vitae nunc sed velit dignissim');

insert into post(createDate,ownerId,ccGroupId,content)
values ('2014-01-01',5,5,'Merry Christmas');
insert into post_like
values
(78,37),
(78,38),
(79,39),
(79,40),
(79,45),
(79,46),
(80,44),
(81,45),
(82,46),
(81,41),
(82,42),
(83,43),
(84,37),
(85,39),
(85,38),
(85,44),
(85,45);

insert into register
values
(28,78),
(29,79),
(30,80),
(31,81);

insert into
    school(location, schoolName)
values
('Florida','University of Florida'),
('New York','Barnard College'),
('Utah','Dixie State University'),
('Alabama','Alabama State University'),
('Utah','Utah State University');

insert into
    user_skills
values
('9', 'python'),
('10', 'research'),
('11', 'react.js'),
('12', 'c++'),
('9', 'java spring'),
('11', 'french');

insert into
    user(
        phoneNumber,
        email,
        uPassword,
        birthday,
        firstName,
        lastName,
        position,
        location,
        headline,
        schoolId,
        startDate
    )
values
    (
        '553452368',
        'greg.shoemaker@hotmail.com',
        'greg236541',
        '1990-12-31',
        'Greg',
        'Shoemaker',
        'Student',
        'Florida',
        '',
        '28',
        '2008-08-25'
    ),
    (
        '553456368',
        'doris90@hotmail.com',
        '23dor6541',
        '1990-04-12',
        'Doris',
        'Ragan',
        'Student',
        'Washington',
        '',
        '29',
        '2008-06-15'
    ),
    (
        '553432368',
        'wheel.arthur@hotmail.com',
        'wheeler236541',
        '1990-11-23',
        'Arthur',
        'Wheeler',
        'Student',
        'Atlanta',
        '',
        '30',
        '2008-03-15'
    ),
    (
        '553452328',
        'anton.delph@hotmail.com',
        'ant236541',
        '1990-06-30',
        'Antonio',
        'Delph',
        'Student',
        'Texas',
        '',
        '30',
        '2008-04-05'
    ),
    (
        '553402368',
        'r.jensen@hotmail.com',
        'jensen3259368',
        '1970-12-31',
        'Ralph',
        'Jensen',
        'Teacher',
        'Florida',
        '',
        '31',
        '2000-08-25'
    ),
    (
        '553381368',
        'finch.75@hotmail.com',
        'fincher3259368',
        '1975-10-21',
        'Mark',
        'Finch',
        'Teacher',
        'Ohio',
        '',
        '32',
        '2002-08-25'
    ),
    (
        '553256368',
        'michele.dom@hotmail.com',
        '47u76568',
        '1978-08-15',
        'Michele',
        'Dominguez',
        'Teacher',
        'Michigan',
        '',
        '28',
        '2005-08-25'
    ),
    (
        '553402236',
        'kasi.brown@hotmail.com',
        'kas436346brown',
        '1971-01-05',
        'Kasi',
        'Brown',
        'Teacher',
        'North Carolina',
        '',
        '31',
        '2001-08-25'
    );

insert into user_work_experiences
values
(82,'4 Year at Tesla'),
(82,'1 Year at Google'),
(79,'6 months internship at Amazon'),
(78,'3 months internship at Amazon'),
(80,'3 months internship at Shell');

INSERT INTO `grade` (`userId`, `gradeId`, `lessonId`, `note`, `perce
nt`, `gTeacherUserId`) VALUES
```

```
(78, 158, 10, 45, 40, 82),
(79, 163, 4, 59, 60, 83),
(81, 187, 3, 84, 60, 85);

INSERT INTO `user_skills` (`userId`, `uSkills`) VALUES
(78, 'python programming'),
(84, 'storytelling'),
(83, 'attention to detail');
```

SELECT Statements

```
select
    name,
    industry,
    headline
from
    company;

select
    *
from
    school
where
    location = "Utah";

select
    firstName,
    lastName,
    position,
    birthday
from
    user
where
    position = "Teacher"
    and birthday > "1972-01-01"
order by
    (firstName) asc;

select
```

```
    c.schoolId,
    c.collegeType,
    s.location
from
    college c,
    school s
where
    c.schoolId = s.schoolId;

select
    l.lessonName,
    l.semester,
    g.note,
    g.percent
from
    lesson l,
    grade g
where
    l.lessonId = g.lessonId;

select
    u.firstName,
    u.lastName,
    u.location,
    t.tRank
from
    user u,
    teacher t
where
    u.userId = t.userId;

select
    u.firstName,
    u.lastName,
    u.location,
    g.note,
    g.percent
from
    user u,
    grade g
where
    u.userId = g.userId;
```

```sql
select
    u.firstName,
    u.lastName,
    g.note,
    g.percent,
    l.lessonName,
    l.semester
from
    user u,
    grade g,
    lesson l
where
    u.userId = g.userId
    AND g.lessonId = l.lessonId;

select
    distinct u.firstName,
    u.lastName,
    g.lessonId,
    g.note
from
    grade g,
    teacher t,
    user u
where
    g.gTeacherUserId = t.userId
    and t.userId = u.userId;

select
    distinct u.firstName,
    u.lastName,
    us.uSkills,
    ue.uEducation
from
    user u,
    user_skills us,
    user_education ue
where
    u.userId = us.userId
    AND us.userId = ue.userId;
```

UPDATE Statements

```
update company
set industry = 'Liquid Fuel'
where company.name = 'Shell';

select * from company;

update job
set jobTitle = 'Region Manager'
where jobId = 1;

update job
set jobTitle = 'Senior Fullstack Web Developer'
where jobId = 4;

UPDATE `grade`
SET note = 90
WHERE lessonId = 158;

UPDATE `school`
SET location = 'New Jersey'
WHERE schoolId = 30;

UPDATE `teacher`
SET tRank = 'Professor'
WHERE userId = 82;
```

DELETE Statements

```
DELETE FROM `grade`
WHERE userId = 78;

DELETE FROM `school`
WHERE schoolName = 'Barnard College';

DELETE FROM `user_interests`
WHERE userId = 82;

UPDATE `lesson`
SET semester = 'semester 5'
```

```
WHERE lessonName = 'Programming Languages';
```

TRIGGERS

```
DROP TRIGGER IF EXISTS `proje`.`join_AFTER_INSERT`;

DELIMITER $$
USE `proje`$$
CREATE DEFINER=`root`@`localhost` TRIGGER `proje`.`join_AFTER_INSERT
` AFTER INSERT ON `join` FOR EACH ROW
BEGIN
    update `group`
    set
        memberCount = memberCount + 1
    where groupId = new.groupId;
END$$
DELIMITER ;

DROP TRIGGER IF EXISTS `proje`.`post like_AFTER_INSERT`;

DELIMITER $$
USE `proje`$$
CREATE DEFINER=`root`@`localhost` TRIGGER `proje`.`post like_AFTER_I
NSERT` AFTER INSERT ON `post_like` FOR EACH ROW
BEGIN
    update post
    set
    likeCount = likeCount+1
    where post.postId = new.postId;
END$$
DELIMITER ;

DROP TRIGGER IF EXISTS `proje`.`school_AFTER_INSERT`;

DELIMITER $$
USE `proje`$$
CREATE DEFINER=`root`@`localhost` TRIGGER `proje`.`school_AFTER_INSE
RT` AFTER INSERT ON `school` FOR EACH ROW
BEGIN
IF new.schoolName LIKE '%University%' THEN
insert into university(schoolId,universityName,universityLocation)
        values(new.schoolId,new.location,new.schoolName);
```

```
END IF;
IF new.schoolName LIKE '%College%' THEN
insert into college(schoolId,collegeName,collegeLocation)
        values(new.schoolId,new.location,new.schoolName);
END IF;
END$$
DELIMITER ;

DROP TRIGGER IF EXISTS `proje`.`school_AFTER_INSERT`;

DELIMITER $$
USE `proje`$$
CREATE DEFINER=`root`@`localhost` TRIGGER `proje`.`school_AFTER_INSE
RT` AFTER INSERT ON `school` FOR EACH ROW
BEGIN
IF new.schoolName LIKE '%University%' THEN
insert into university(schoolId,universityName,universityLocation)
        values(new.schoolId,new.location,new.schoolName);
END IF;
IF new.schoolName LIKE '%College%' THEN
insert into college(schoolId,collegeName,collegeLocation)
        values(new.schoolId,new.location,new.schoolName);
END IF;
END$$
DELIMITER ;
```

CHECK Constraints

```
ALTER TABLE `student`
ADD CONSTRAINT CHCKGPA CHECK (gpa>=0);

ALTER TABLE `user`
ADD CONSTRAINT CHCKID CHECK (userId>=10);

ALTER TABLE `comment`
ADD CONSTRAINT CHCKIDS CHECK (postId>=0 AND commentId>=0 AND
wUserId>=10);


ALTER TABLE `comment`
ADD CONSTRAINT CHCKIDS CHECK (postId>=0 AND commentId>=0 AND
wUserId>=10);
```