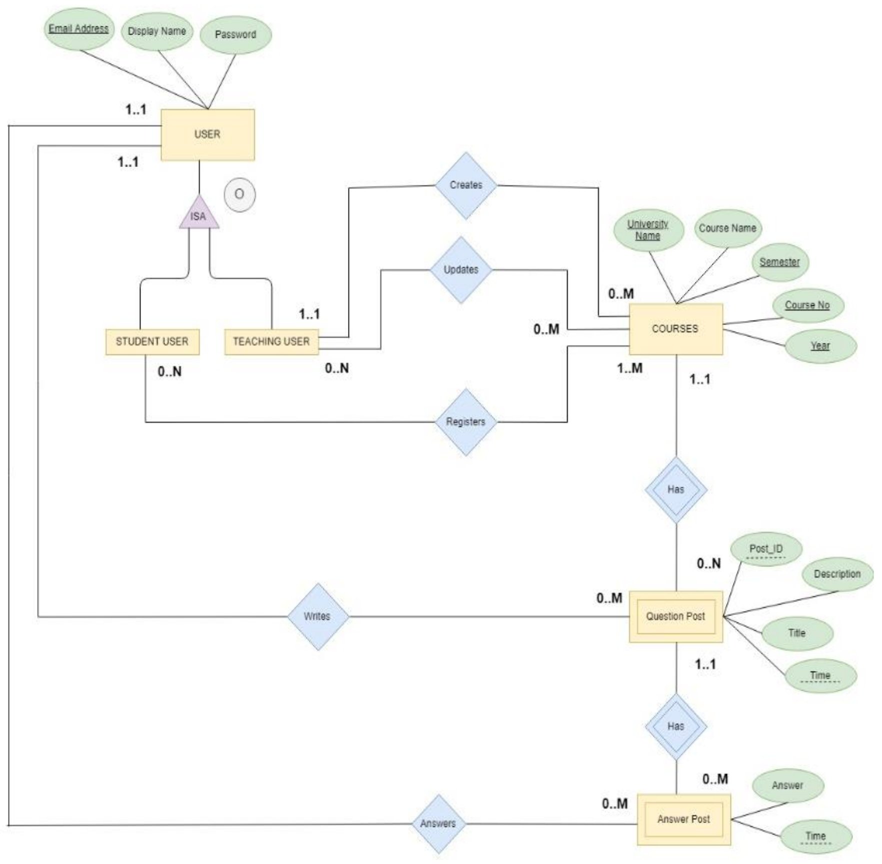
Entity-Relationship Diagram
Ganesh Ezhilan
CSE-560
Project - 1, Tiny Piazza

E-R Diagram:



E-R Model Introduction

In my “Tiny Piazza” model I have four entities with two subclasses among them and each entity has different attributes which are

Entities and Attributes:

1.**Users** – Email ID, Display Name, Password.
In this entity Email-Id will be my primary key and display name and password are my other attributes. These attributes satisfy the condition that each email-id should have only one display name since email-id is primary key. The Users entity is a strong entity since it does not depend upon any other entity.

This Users entity have two subclasses named "**StudentUser**" and "**TeachingUser**" which can be formed by ISA relationship (I.e. specialization) from super class entity Users. These subclasses follow **total over-lapping** participation.

2.**Courses** – <u>University Name</u>, <u>Semester</u>, <u>year</u>, <u>course Number</u>, course Name.
In this entity there are five attributes of which four join together to form the key attributes of the **Courses** entity which means the rows in courses entity can be uniquely identified with those four attributes

3.**QuestionPost** - <u>University Name, Semester, year, course Number, Post-ID</u>, Description, Title, Time.
This is one of the weak entity since its existence dependent upon another strong entity **Courses**. We can't represent the each of the rows uniquely in **QuestionPost** entity without the help of the Courses entity so it forms weak entity with courses. This entity also depends upon another strong entity **Users** but can be represented uniquely without the help of **Users** entity so it is not weak entity of **Users**

4.**AnswerPost** - <u>University Name, Semester, year, course Number, Post-ID</u>, Answer, Time.
This is one of the weak entity since it is dependent upon another entity question Post for its existence and also we can't represent each rows uniquely in "**AnswerPost**" entity without the help of another entity so it forms weak entity with "**QuestionPost**".

User-Management:
These entity comes under the user-management consist of all the users of tiny piazza (i.e. teachers, teaching assistants, students). It has three attributes as mentioned early which are email-id, display name and password. Users can login with the help of Email ID and password. Since email-id is primary-key attribute it can take care the constrain that each email-id can have only one name and password. Tiny Piazza manages two types of subclasses **TeachingUser** and studentUser which is the specialization of the entity **Users**. The primary key is **Email-ID**

Course-Management:
These have five attributes of which four are key attributes which are University Name, Semester, year, course Number and non-key attribute course name. All of the attributes are non-null. By which it stores all the course information for different university for different years.

Post-Management:
In these system I had divided the post type entity types into two which are "**QuestionPost**" entity and "**AnswerPost**" entity. QuestionPost entity will be taking care of all the question post posted by each individual user of Tiny Piazza which means user can be student, Teaching assistants and Instructor as well. The Post-id will be partial key which means each question can be identified uniquely with the help of **postId** and courses key attributes. In "**AnswerPost**" the time attribute will be partial key. Both of these entities are weak entities since they can't exist without the help another entity. All the attributes in these entities are non-null

<u>Cardinality Constrains</u>

User-Course Relation:

Relationship between **Users** and **Courses** is connected through the relation “Register” and “creates/updates”. After specialization **Users** can be spilt into two which are StudentUser and **Teachinguser**. Based upon the description only “**TeachingUser**” can able to create/update the courses and StudentUser can only be able to register for different courses. Here I kept the cardinality from student user to course as 1:M because of the thing that each student should enroll at least for one course to maximum of n number of courses similarly each course can have 0 to M number of student. For TeachingUser entity I had made the cardinality such that each **TeachingUser** can create many courses but each course can be created by only one **Teachinguser** so it is 1:M relation and for updating each course it is N:M users which means many teachingusers can update 0 to many courses.

Course-Post Relation:

The relationship between courses and question post is connected through the relation “has”. It has the cardinality of 1:M such that each question can only come under one courses but each course can have zero to many questions from different users so relationship between them can be given as 1: M.

Question-Answer Relation:

The AnswerPost is connected to QuestionPost through “has” relation. It has cardinality of 1:M. Here each question can have zero to many Solution/Answers while, each answer can have only one question for a particular course so it is one to many relations

User-Question Relation:

The relationship between them can be given as 1:M because each question under one can be created by only one user which means one question under a course cannot be created by multiple user and each user can create zero to many questions so relation is one to many

User-Question Relation:

The relationship between them can be given as 1:M because each answer can be written by only one user and each user can write multiple answers for a single question so the relationship is one to many.

Mapping E/R diagram to Relational schema:

USER-MANAGEMENT

Users Table:

The users table has the data of all the users who have signed up to tiny piazza. The table has attributes EmailId, DisplayName and UserPassword. The attribute email Id is the primary key here as any user can be uniquely identified by it. All the attributes have type varchar since all are strings.

Course Table:

The course table has the data of all the courses for different universities. The table has attributes University Name, Semester, year, course Number, and course Name. The attributes UniversityName, Semester, CourseYear, courseNumber are all combine together to form the primary key for this table only the year attributes are integer type. All other attributes are varchar type.

Teaching User Table:
Since we have created subclasses instructor and student to our superclass which is the specialization for the users, we now have tables for teachingUser which have instructor's Email ID. These email ids are now primary keys for table as they are unique. A user who is a teaching assistant will also be present in in this table. Queries related to teachers only or students only can be executed quickly. This table can be used for fetching the Email-IDs of the teaching user during creation and updation of courses by the user.

COURSE-MANAGEMENT

course Create Table:
It has the course attributes which form the course id and also the email id of the instructor who has created the course. The email id is a foreign key which is referred from the instructor table.
This puts a constraint that the course can only be created by a teaching user. Setting the course primary key attributes of courses makes sure that the course id cannot be repeated and a course can only be created by one teaching user. Here I'm not setting Email-Id as primary-id as primary key because creating as primary key will violate the constrain that course can be updated by any one from the Teaching User table.

course Register Table:
This table has all the primary key attributes of courses and email ids of all the users who have registered for the course including the User who has created the course. I had given the permission for the teaching user can even enroll in other course for example TA can be a student for other course as well. To ensure this the course id is referred to the courseCreate table which means that the course should be created before the user register for a particular course and the email ids are referred to the Users table email id, which means all the user who have signed up can register for the course (i.e. Student, Instructor and Teaching Assistant)

Course Update Table:
This table is used to keep track of the teaching user who can able to update the courses. This table has all the primary key attributes of the courses table and the attribute email id which has been referred from the course table. This is to enforce that only the course instructors (all of them) can update the course but not all the instructors who have signed up on tiny piazza. There is no primary key specified here because the teaching user can update the course any number of times not only the user who created the course can update.

POST-MANAGEMENT

Question Post Table:
The table stores all the values of the questionPost posted by the different user. The table has all the course attributes, emailId of all the users who posted the question and also questionPost attributes like post id, title, time and description in which postId is the partial key. The course id and email id are foreign keys referring to the course members table, which means that only the course members (student and instructors) registered to the course can post questions. The primary keys of the QuestionPost table are all the foreign keys along with the partial key PostId. The email id of the user who creates question is included in the primary key to put a constrain that the post can be updated only by that user who created post.

Answer Post Table:
This table consists of all the answers for different questions under different courses. It has two attributes time and answer. I assuming time as my partial key since no two users can post the answer at the same time. The answers have key attributes of question post as both foreign and primary along with its own partial key time which means that there can be multiple answers to a question post by the same user or multiple users. Here I removed Email-Id so that all the can able to answer for different questions only when they are registered for that particular course.

Further Discussion

Advantages:

1.Only the user who created the question post can able to delete or update the particular post
2.Only the user who enrolled for a particular course can able to ask question or post answer for the question for a particular course
3.The TA can even able to enroll for other courses for which he/she is not teaching assistant
4.Specialization for different types of users were handled perfectly through total over lapping participation
5.Reduced the work load for the application developer by handling all the cases of constrains and uniqueness so that application developer no need to write many functions to satisfy different constrains

Disadvantages:

1.The representation of same attributes for different tables fact take more space when the data is stored.
2.The password field should not be kept open since password for different user can be seen by persons who is handling database.
3.Many redundancies in the table creation since there are many tables with same attributes.
4.We can’t able to differentiate between who posted the answer (i.e. whether the question was answered by instructor or student)

5.Extra-Credits question

Yes, after decomposition it preserves dependencies. Since R1 union R2 gives us the same relation {ABCD} (i.e. it renders functional dependencies) also intersection of R1 and R2 won't be giving us the null value instead it returning {AB}. So after decomposition it contains the candidate keys so we can say that decomposition is lossless. So we can say R1 and R2 are in BCNF