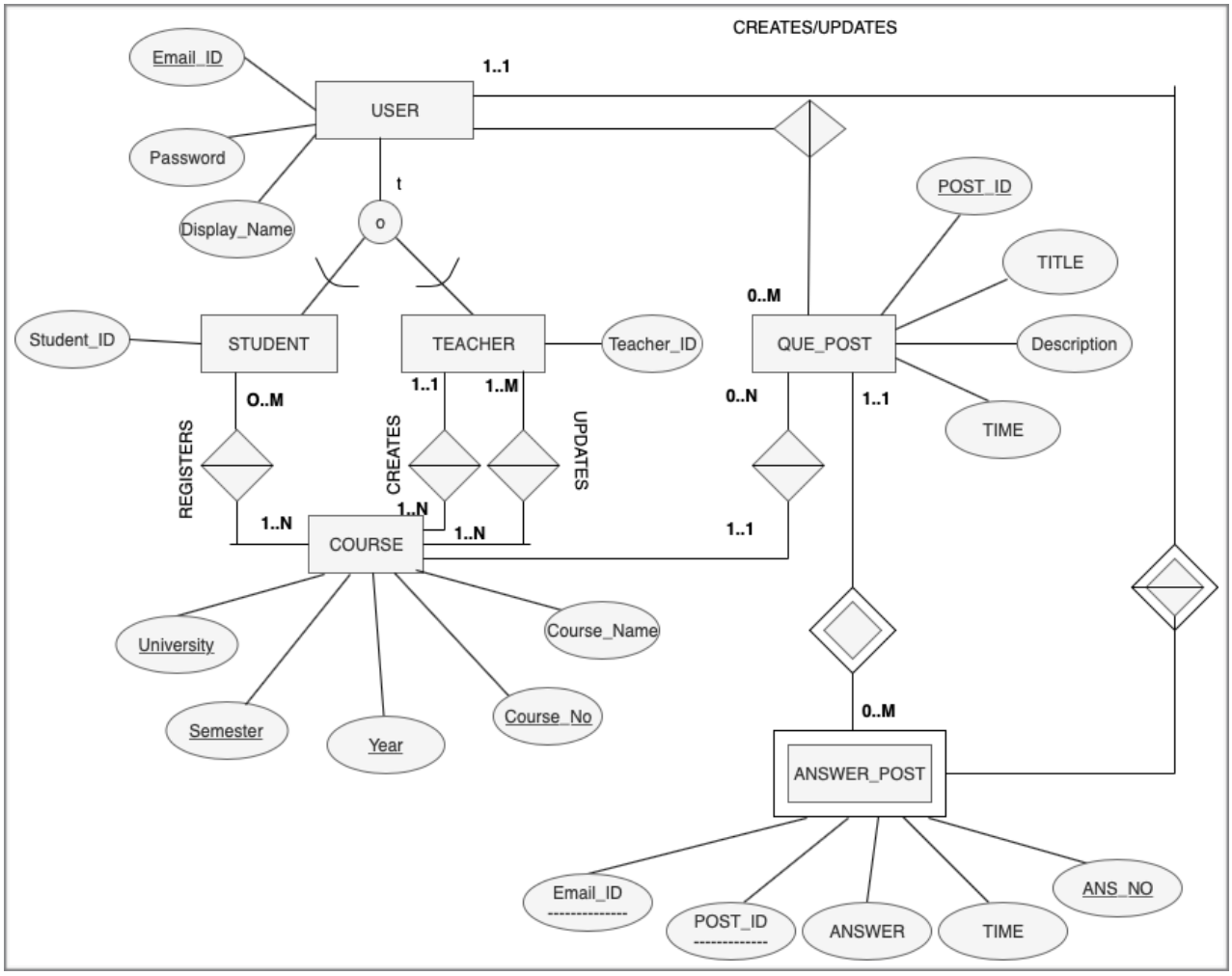
**PROJECT 1**
**CSE460LR000:Data Models and Query Language**

Submitted By: NUPUR SUNIL AGRAWAL

We were to design and implement the database schema for TinyPiazza, which is a course forum website provides simple functions, main functions of TinyPiazza are as the following:
• User management: user sign up, user login/logout.
• Course management: course basic information.
• Post management: managing posts in courses.
• User-Course relationship management: users can create/update or register courses.
• User-Post relationship management: users can create/update/delete posts.

## E/R Schema:

The E/R model for this was plotted as followed:



**User management :**

- A user signs in using their email-address, a username and password.
- An entity **USER** was created which has 3 attributes - **Email_ID, password, Display_name.**
- A total specialization was made for **USER** since all the users are included in either students or teachers, or both where **USER** is the superclass and **STUDENT** and **TEACHER** are the subclasses.
- An overlap specialization was made since a user can be a student as well as an teacher eg: Teaching Assistant.
- The **STUDENT** entity has an attribute **Student_ID.**
- The TEACHER entity has an attribute **Teacher_ID**.

**Course Management:**

- An entity **COURSE** was made which had the following 5 attributes: **University, Semester, Year, Course_No and Course_Name**.
- **University, Semester, Year** and **Course_No** together form the primary key of **COURSE**.

**Post Management:**

- Users can post questions within a course and answer questions too.
- An entity **QUESTION_POST** was created which consisted of the following 5 attributes Post_ID, Title, Description, Time.
- It has a week entity **ANSWER_POST** attached to it which consists of the following attributes: **Email_ID, Post_ID, ANSWER, TIME, ANSWER_NO**.
- The inherited keys Email_ID and and Post_ID together with the Answer_No form the primary key of the **ANSWER_POST.**

**User-Course relationship management:**

- The subclasses STUDENT and TEACHER of superclass USER are related to COURSE.
- Since, 1 student can register for 1 to N courses the relation has cardinality **(1..N)**.
- Since a course can be registered by 0 to M students the relation has cardinality **(0..M)**.
- Since a Teacher can create 1 to N courses it has cardinality (**1..N**) and a course can be created by exactly 1 teacher it has cardinality (**1..1**).
- Since a teacher can update many courses it has cardinality (**1..N**) and a course can be updated by many teachers it has cardinality (**1..M**).

**User-Post relationship management:**

- A user can post/update many courses so the user to question post relationship has cardinality (**0..M**) but a post can be created by exactly 1 user so that has cardinality(**1..1**).
- A course can have many posts within it so it has cardinality (**0..M**) but a question post can belong to exactly 1 course so it has cardinality (**1..1**).
- The answer posts can have exactly 1 question so the cardinality is (**1..1**) but a question can have many answers so the cardinality is (**0..M**).

## Relational Database Schema:The schema consists of the following tables:

- USER
- STUDENT
- TEACHER
- COURSE
- QUESTION_POST
- ANSWER_POST
- REGISTERS
- CREATES
- UPDATES

**USER**

- The table User consists of 3 columns Email_ID, username and password.
- Email_ID is the primary key and the username is unique.

**STUDENT**

- The table Student consists of 2 columns student_ID and Email_ID.
- Here Student_ID is the primary key and email_ID is the foreign key from the table USER.

**TEACHER**

- The table Teacher consists of the columns teacher_ID and Email_ID where Teacher_ID is the primary and email_ID is the foreign key from the table USER.
- This table consists the list of all the teachers and their teacher_ID.

**COURSE**

- The table course consists of course details of all the courses.
- It has the following 5 columns course_ID, school_year, semester, university and course_name.

**QUESTION_POST**

- The table QUESTION_POST consists of all the questions within a course so it consists of foreign keys course_ID, school_year, semester, university establishing a relationship between the question_post and the course table.
- It also consists of the user_id which is a foreign key referring to table USER thus establishing a relationship between the user and the question_post.
- The other attributes are timestamp, title ,description and post_id (primary key).

**ANSWER_POST**

- The table ANSWER_POST has primary keys answer_no post_ID and User_id which are foreign keys referring to the tables question_post and USER. Thus we have established a relationship between user and answer post and question_post and answer_post.

- It also consists of columns answer and timestamp.

REGISTERS
- This table establishes a relationship between the student and the course.
- It consists of student_id and the course_id which together make the primary key of this table.

CREATES
- This table establishes a relationship between the teacher who created the course and the course.
- It consists of the course_id which is the primary key and teacher id which is a foreign key from the table TEACHER.

UPDATES
- A course can be updated by multiple teachers. This table establishes a relationship between the teachers who updated a particular course.
- The teacher_id and the course_id together form the primary key of this table.

Advantages:
The schema is normalized thus avoiding redundancy. The model consists of an answer_no which allows us to search for a particular answer using the key. The answer_post is related to user and the question post thus inheriting their primary keys and letting us know who has answered the question.

Disadvantages:
Some of the functionaries of the E/R diagram are lost while converting it to a relational database. For example, we cannot put certain constraints which are expected in the model in our database schema.

# Q.5

Yes they are in BCNF.

A→BC: Although the value of A determines the values of B and C, we cannot infer their values from other tuples because no two tuples in R have the same value for A which implies no redundancy.

A→BD: Although the value of A determines the values of B and D, we cannot infer their values from other tuples because no two tuples in R have the same value for A which implies no redundancy. Thus all decendencies are preserved.