CSE 560 Project1: TinyHub

Pratibhaa Reddy Kandimalla

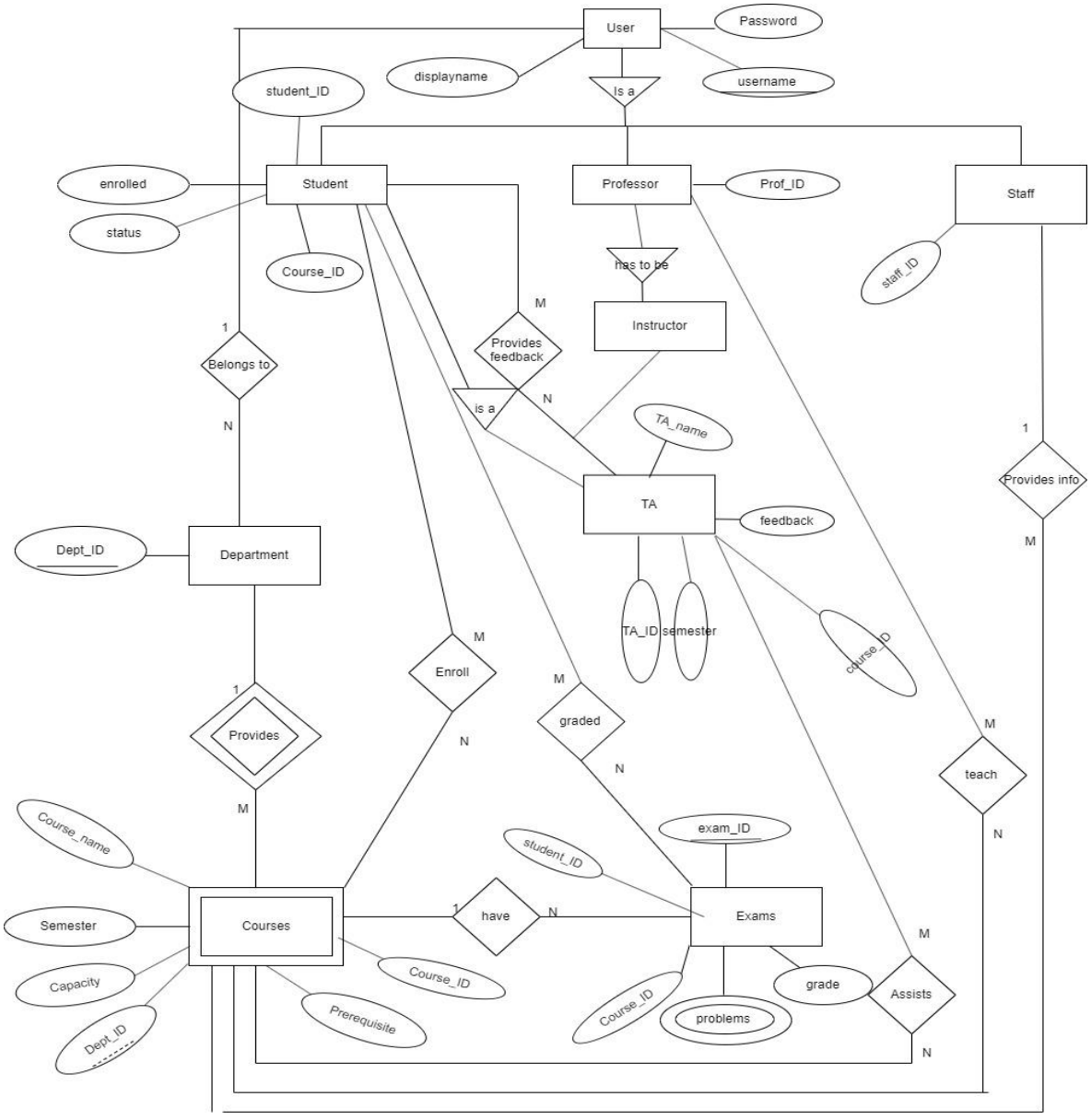
UB ID: 50299669

E/R Schema:

Design and implement the database schema for TinyHub, which is a course enrollment website. It provides the below functions:

1. User-management
2. Department-course relationship management
3. Enrollment

Below is the E/R diagram for TinyHub:



List of Entities:

1. User
2. Student
3. Professor
4. Staff
5. Department
6. Courses
7. Exams
8. Instructor
9. TA

Attributes for an Entity:

1. **User:**
    - username (Primary Key) - unique
    - password
    - displayname - unique
2. **Student**:
    - student_ID – To indicate the student using ID
    - course_ID – To specify the course number
    - enrolled – To specify whether a student is enrolled
    - status – To specify the status of a course
3. **Professor:**
    - Prof_ID – To specify the professor using ID
4. **Staff**:
    - Staff_ID – To specify the staff using ID/ number.
5. **Department**:
    - Dept_ID – To identify the department using the number (unique & not null)
6. **Courses**:
    - Course_ID
    - Course_name
    - Dept_ID
    - Capacity
    - Semester
    - Prerequisite
7. **Exams**:
    - Exam_ID
    - Student_ID
    - Course_ID
    - Grade
    - Problems – It is a multivalued attribute which consists of <Problem number, Score> values in an exam
8. **Instructor**:
    - Prof_ID
9. **TA**:
    - TA_ID

- TA_name
- Course_ID
- Semester
- Feedback

Cardinalities between the Entities:

1. User <Belongs to> Department- **1: N**
2. Department <Provides> Courses- **1:M**
3. Courses <have> Exams- **1: N**
4. Students<Provide Feedback> TA- **M: N**
5. Students <Provide Feedback> Instructor- **M: N**
6. Staff <Provides Info> courses- **1:M**
7. Professor <teach> courses- **M: N**
8. Student <enroll> courses- **M: N**
9. Student <graded> Exam- **M: N**
10. TA<Assists> courses- **M: N**
11. User **IS A** Student / Professor/ Staff
12. TA **IS A** Student

Requirements for User & how it satisfies

- User is a student/ Professor/ Staff: User <IS A> student/ Professor/ Staff
- User belongs to Department: This condition will be satisfied by using the relationship, User <belongs to> Department

Requirements for Student & how it satisfies

- Student can have multiple majors: Student<Belongs to> Department
- Student can enroll a course
    1. If he passes all the prerequisites. This condition can be satisfied by checking the 'enrolled' attribute (from student database), which gives us information about whether the student has enrolled into a course or not (either yes or no).
    2. If the course is being offered by a department they are majoring in: This condition can be satisfied by checking dept_ID of a student.
    3. If the capacity of the course is not full: This condition can be checked by using the attribute 'Capacity' which gives information about the course capacity.
- Student will have grades A/B/C/D/F with course, which will be given after the student finishes the course: This condition will be satisfied by using the attribute 'grade'. To check whether the student has finished the course, use attribute 'status'. Once the status is completed, then the student will be given a grade.
- Students can give feedback for instructor/ TAs of the course in which they are enrolled: This condition can be satisfied by the relation ship <Provides Feedback>. Using TA_ID & Course_ID, a student can give a feedback to TA/Instructor. Here, 'feedback' attribute is used.
- Each course has one or more exams which satisfies the relationship: Course <have> Exams- **1: N**

- Each exam has a number of problems. Student have scores on those problems: This condition can be satisfied by using a multivalued attribute, ‘Problems’ which gives the values for each problem ID and its score.

Requirements for TA & how it satisfies:

- All TAS must be students: TA <IS A > Student

Requirements for Courses & how it satisfies:

- Course has a unique number, name, department of the course, instructor, TAs, Prerequisites
- Course can have different TAs and different TAs in different semester: This condition can be satisfied by using the attribute ‘Semester’, ‘TA_ID’, ‘course_ID’. Here the course is a weak entity. Hence Course can’t have a primary key.

Requirements for Staff & how it satisfies:

- Staff provides information about the course in each semester: This condition can be satisfied using the relation, Staff <Provides Info> Course. Here, semester & Course_ID attributes are used.

Requirements for Instructor & how it satisfies:

- Instructor has to be a professor: This condition can be satisfied by using the relation, Instructor <Has to > Professor. Here, the instructor ID is self-recursive to the professor ID.

Requirements for Department & how it satisfies:

- Department has an identifying department number. Hence Dept_ID is primary key for Department Entity.

Advantages of the above Design:

- It satisfies all the above-mentioned requirements.

Disadvantages of the above Design:

- It might cause some redundancy of data. For example, if we join 2 tables (Student & Course), there might be some extra records of data which don’t add value to the schema and causes some redundancy.