CSC 540 (001)
Database Management Concepts and Systems
Project - 1

Louis Le (lle3)
Chris Brown (dcbrow10)
Abhimanyu Kumar (akumar24)

ER DIAGRAM

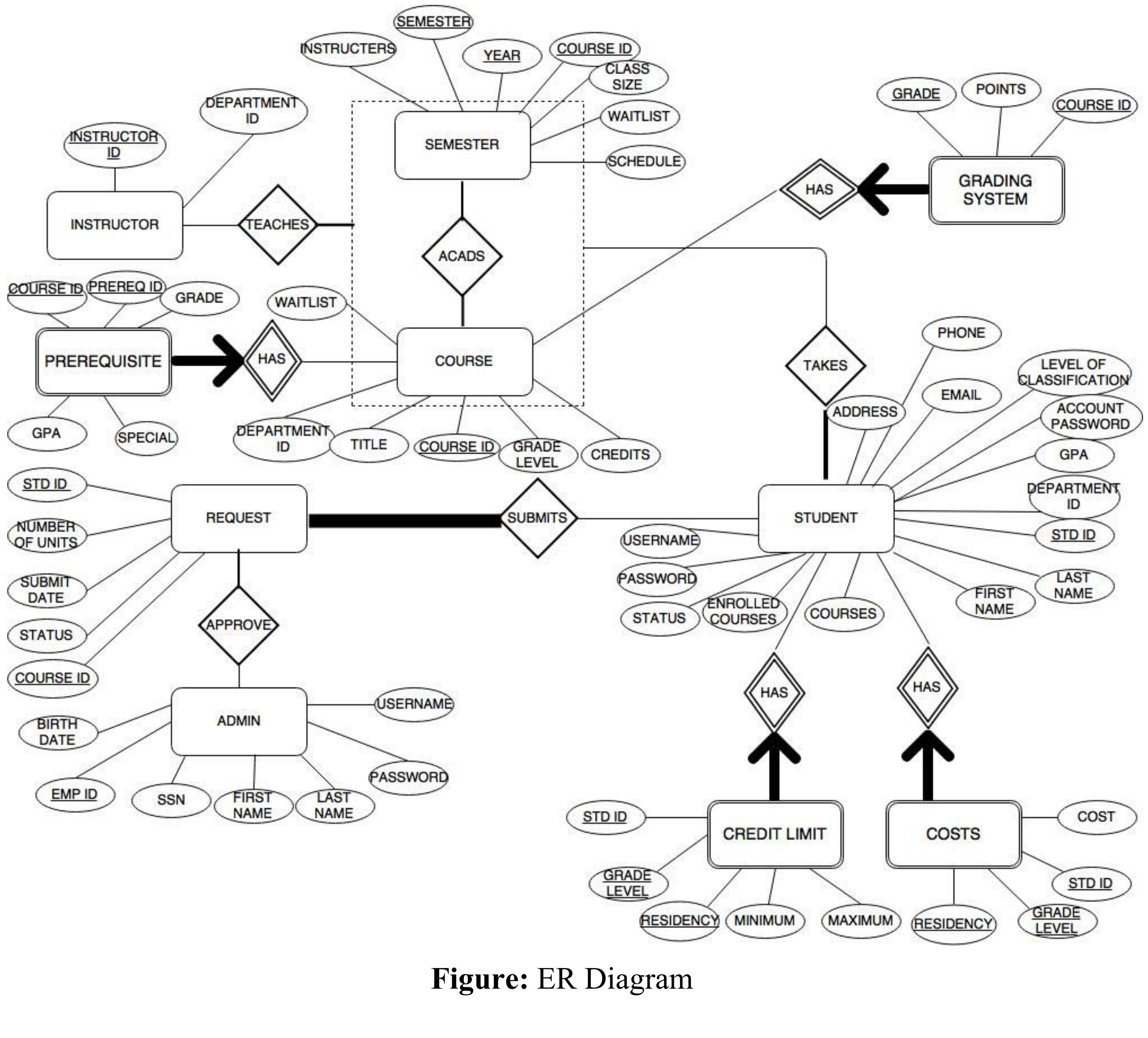

Figure: ER Diagram

- **Entities:** Admin, Request, Student, Course, Prerequisite, Semester, Instructor, Costs, Credit Limit, Grading System
- **Aggregation Relationship:** Course and Semester
- **Binary Relationships:** Admin-<Approves>-Request, Student-<Submits>-Request, Student-<Takes>-[Course/Info/Semester], Instructor-<Teaches>-[Course/Info/Semester]
- **Weak Entities:** Student: Costs, Credit Limit; Course: Prerequisite, Grading System

Description of each entity is done under the section "Relational Model".

CONSTRAINTS

1. **DATABASE CONSTRAINTS**

   This section involves the constraints that couldn't be implemented in the database but had to be implemented in the application code.

   a. Check if class is full to add student to course waitlist and add student from waitlist to class when an enrolled student drops.
   b. Student wanting to add/remove courses on a semester. One needs to check whether the student is still within the minimum and maximum credits required depending on their classification
   c. Add/drop date for adding/removing course from semester
   d. Checking to see if student has met all the prerequisites are met when enrolling for a course
   e. Deleting a request after it has been approved/rejected
   f. Calculate a student's updated GPA
   g. Making sure that students and teachers don't have courses that take place at the same time
   h. Admins can check if student has paid bill before adding to course or waitlist.
   i. Adjust student bills to reflect changes in schedule
   j. If the status for the request is approved/rejected, the who and date is not automatically reflected.
   k. If a student tries to enroll in a courses outside his/her classification, an automatic request has to be generated for permission from the department offering the course.

2. **NORMAL FORMS**

   We have used the First Normal Form (1NF) in our project. We have used 1NF because there weren't many functional dependencies and the ones which were there were mainly because of the primary key attribute.

RELATIONAL MODEL

1. **ADMINISTRATOR:** This table contains data of all the administrators. The admin will approve requests submitted by the students for adding a course in their time table. The admin is also responsible for adding courses and students entries in the university database. The referential constraints involved are:
    a. Course ID (from Course)
    b. Student ID (from Student)
    c. Semester (from Semester)

    The functional dependencies involved are:
    a. ADMIN.EMP_ID -> ADMIN.SSN, ADMIN.FIRST_NAME, ADMIN.LAST_NAME, ADMIN.PASSWORD, ADMIN.USERNAME
    b. ADMIN.SSN -> ADMIN.FIRST_NAME, ADMIN.LAST_NAME, ADMIN.PASSWORD, ADMIN.USERNAME
    c. ADMIN.USERNAME -> ADMIN.FIRST_NAME, ADMIN.LAST_NAME, ADMIN.PASSWORD, ADMIN.EMP_ID, ADMIN.SSN

2. **COURSE:** This table has the data for all the courses taught in a particular semester. The data even involves the location, schedule, number of waitlists of a particular course. Only the admin has the authority to edit this table. The students will only be able to view it and add courses for their semester term. Some of courses come with a prerequisites, relational entity of which is a weak entity to the courses relational entity. The are no referential constraints involved.
    The functional dependencies involved are:
    a. COURSE.COURSE_ID -> COURSE.DEPT_ID, COURSE.TITLE, COURSE.SIZE, COURSE.CREDITS, COURSE.NUMBER_UNITS, COURSE.MAXIMUM_SEATS, COURSE.WAITLIST, COURSE.MAXIMUM_WAITLIST, COURSE.LOCATION, COURSE.SCHEDULE

3. **INSTRUCTOR:** This table includes the list of all the instructors teaching a course in a particular semester. Every instructor is from a particular department of the university. There are no referential constraints involved.
    The functional dependencies involved are:
    a. INSTRUCTOR.INSTRUCTOR_ID -> INSTRUCTOR.DEPT_ID

4. **STUDENT:** Student table has the data for all the students studying at the university. There are no referential constraints involved.
   The functional dependencies involved are:
   a. STUDENT.STD_ID -> STUDENT.FIRST_NAME, STUDENT.LAST_NAME, STUDENT.DEPT_ID, GPA, STUDENT.PASSWORD, STUDENT.CLASSIFICATION, STUDENT.EMAIL, STUDENT.USERNAME, STUDENT.PHONE
   b. STUDENT.EMAIL -> STD_ID, STUDENT.FIRST_NAME, STUDENT.LAST_NAME, STUDENT.DEPT_ID, GPA, STUDENT.PASSWORD, CLASSIFICATION, STUDENT.PHONE, STUDENT.USERNAME
   c. STUDENT.PHONE -> STD_ID, STUDENT.FIRST_NAME, STUDENT.LAST_NAME, STUDENT.DEPT_ID, GPA, STUDENT.PASSWORD, CLASSIFICATION, STUDENT.USERNAME
   d. STUDENT.USERNAME -> STUDENT.STD_ID, STUDENT.FIRST_NAME, STUDENT.LAST_NAME, STUDENT.DEPT_ID, GPA, STUDENT.PASSWORD, CLASSIFICATION, STUDENT.PHONE

5. **CREDIT LIMIT:** This table helps in assigning maximum and minimum credits to a student, depending upon his/her residency status and grade level. Since there will no minimum and maximum credit limit for a student if the student itself is not there, this is a weak entity for the Student table. There is one referential constraints involved: Student ID.
   The functional dependencies involved are:
   a. CREDIT_LIMIT.GRADE_LEVEL, CREDIT_LIMIT.RESIDENCY, CREDIT_LIMIT.STD_ID -> CREDIT_LIMIT.MINIMUM, CREDIT_LIMIT.MAXIMUM

6. **COSTS:** This table stores the bill for each student, if any is there. Since there will no bills due for a student if the student itself is not there, this is a weak entity for the Student table. There is one referential constraints involved: Student ID.
   The functional dependencies involved are:
   a. COSTS.STD_ID, COSTS.RESIDENCY, COSTS.GRADE_LEVEL -> COSTS.COST

7. **GRADING SYSTEM:** This table maps the grades with the points, which will help in calculating the GPA of the student. Since there will be no grades for a

course if the course itself is not there, this is weak entity for the Course table. There is one referential constraints involved: Course ID.
The functional dependencies involved are:

a. GRADING_SYSTEM.COURSE_ID, GRADING_SYSTEM.GRADE -> GRADING_SYSTEM.POINTS

8. **PREREQUISITE:** This table shows the prerequisites of a course. It also includes the special permission required from the department in order for a student to enroll in a course. Since there will be no prerequisites if there is no course, this is a weak relational entity of the Course entity. There is one referential constraints involved: Course ID.
The functional dependencies involved are:
   a. PREREQ.PREREQ_ID -> PREREQ.COURSE_ID, PREREQ.GRADE, PREREQ.PERMISSION, PREREQ.GPA
   b. PREREQUISITE.COURSE_ID, PREREQUISITE.GRADE, PREREQUISITE.GPA -> PREREQUISITE.PERMISSION_FROM_THE_DEPARTMENT
   c. PREREQUISITE.COURSE_ID -> PREREQUISITE.GRADE

9. **REQUEST:** This table involves the request the students will send to the administrator in order to get a course added in their schedule. The referential constraints involved are:
   a. Student ID (from Student)
   b. Course ID (from Course)

   The functional dependencies involved are:
   a. REQUEST.STD_ID, REQUEST.COURSE_ID -> REQUEST.NUMBER_OF_UNITS, REQUEST.DATE, REQUEST.STATUS

10. **SEMESTER:** This table lists the schedule for an entire semester, including the schedule of the classes, which instructor is teaching which course, and many more. The referential constraint involved is: Course ID (from Course).

    The following are the functional dependencies involved:
    a. SEMESTER.SEMESTER, SEMESTER.YEAR -> SEMESTER.ADD_DATE, SEMESTER.DROP_DATE