Table of Contents

Introduction

The figure 1.1 shows the present schema design of the college database which illustrates CREATE TABLE fragments.

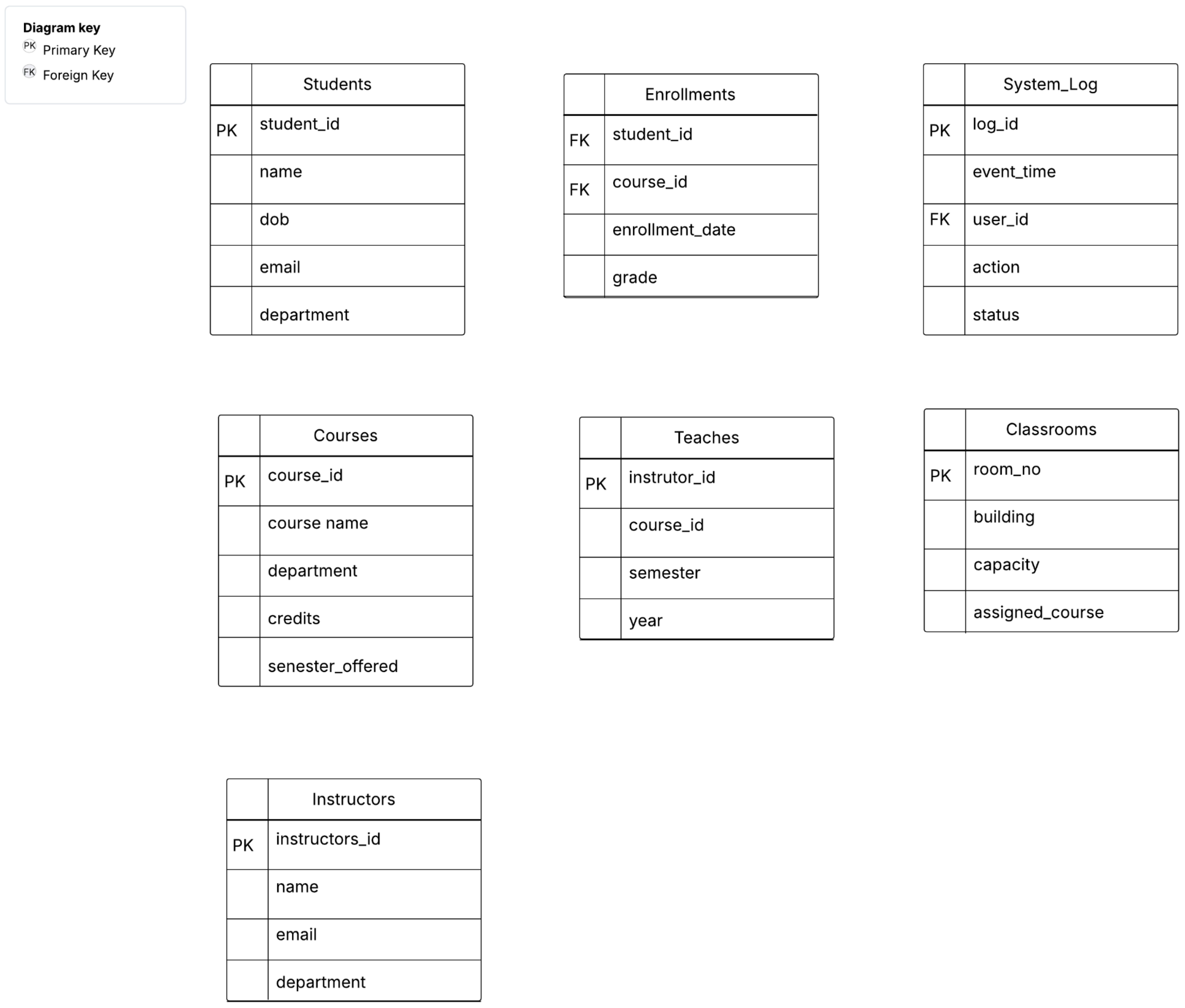


Figure 1.1

This paper critically analyzes the key flaws,redesigned schema with ER diagram,a rationale for each change and recommended indexes,operational controls and constraints.The objective is to improvise data integrity,normalisation,scalability,maintenance and security.

Flaws,Risks and Performance Issue

1. In the classroom entity there are assigned courses but not a Foreign key.There are no foreign key constraints ,so there is no automatic enforcement of relationships between students,courses and enrollment meaning a student could be enrolled in a non-existent course or a course could be deleted while enrollment still references it.

2. The lack of indexes in representing foreign key fields such as student_id or course_id(i have shown in the figure 1.1,but it isn't present in the schema)which can cause severe query inefficiency as the database(DB) enhances.Query performance experience unauthenticated.

3. Column constraints are missing mandatory fields such as email,grade or department allowing NULL or invalid entries that could lead to inconsistent records.Moreover there are no constraints for domain integrity , the problem is no NULL,UNIQUE(email),CHECK constraints.

4. Lack of normalisation could lead to repeated or ambiguous attributes,Department stored as free text in multiple tables(students,courses,instructor).The string semester_offered is vague.

5. because it doesn't show or lead to the specific entity it belongs to or the real value of the attribute in the table.Classroom stores assigned_courses as a single INT which assumes one course is for one classroom permanently.

6. Ambiguous user auditing and security issues in System_log,where System_Log.user_id INT doesn't incorporate proper definition of who is the user (student,instructor or admin) and no foreign key to a user table.

7. Classrooms.assigned_course suggests only one course can be assigned there isn't proper scheduling of the time slot,timetable or no handling of recurring classes which shows the incompleteness of the DB.

8. Weak logging system doesn't have reference to user_d and also doesn't link it to a user table,and has unclear representation of who the user is(student,admin or instructor).

Overall, this schema design could lead to maintenance,data integrity and query performance making it unsuitable for large scale environment significant modification.

Redesigned DB Schema shown using the ER model

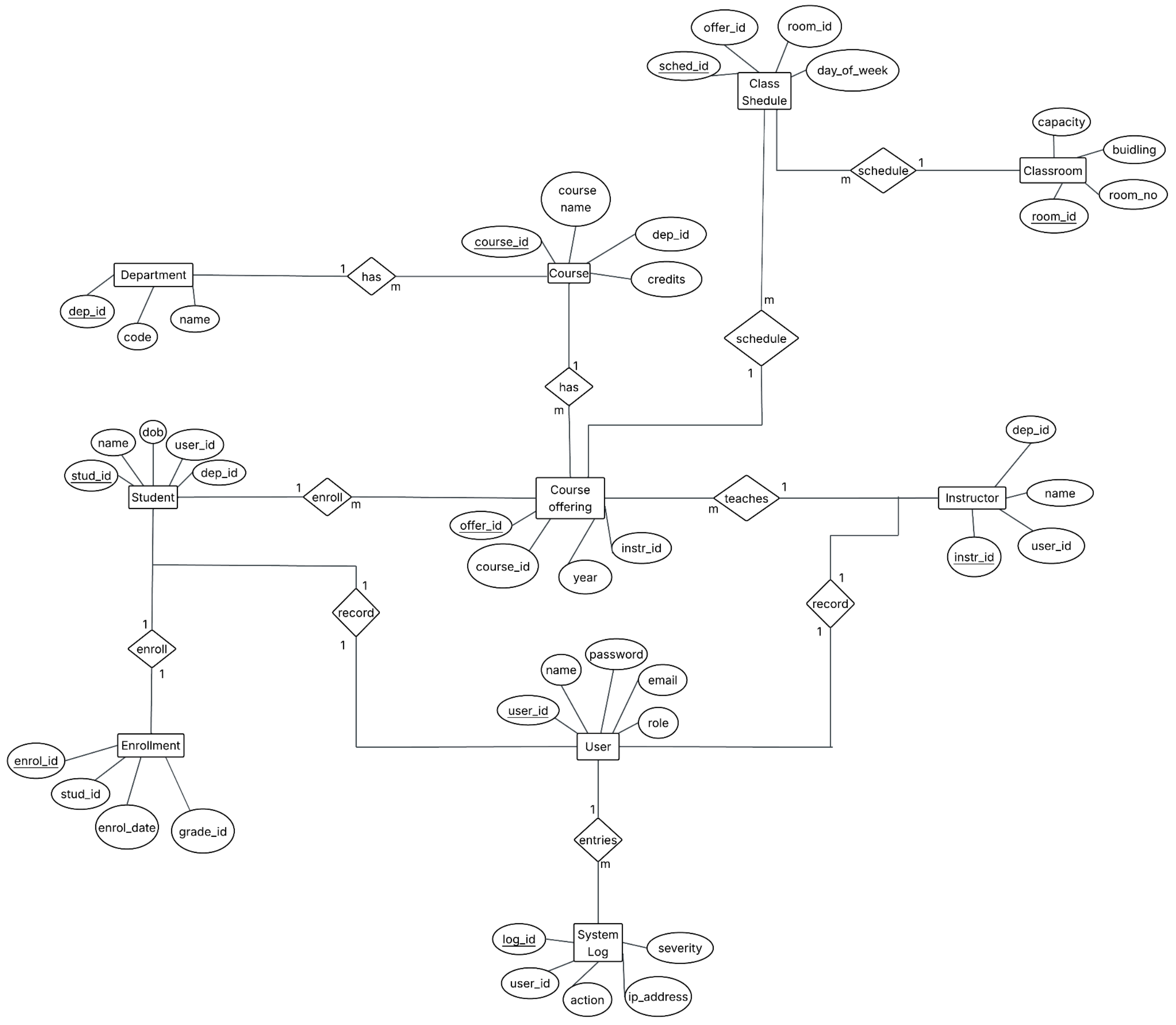


Figure 1.2

Explanation

This ER diagram shows the designed schema of the SQL for the university management system.The redesigned ER diagram represents the main entities and attributes needed to be present for the university database.Shown class schedules help in representing the time table slots.Courses offered has relationship with student and teacher showing which particular course is taught by particular teacher , whereas students are enrolled in the particular courses.Both the teacher and student entity is linked with user entity to so there is no confusion with who is the user that can log in to the system.

Conclusion

The redesigned database schema greatly improves addressing the original schema's fundamental flaws such as missing foreign key constraints,poor normalisation,weak constraints and incomplete scheduling .The redesigned schema addresses these critical issues by enforcing integrity,normalising data structure such as departments,introducing strong scheduling mechanisms and improving operational controls which shows clear user roles and audit trails.Overall the redesign creates a maintainable and efficient database capable of scaling with academic institution needs while prioritising data safety,quality and security.

Appendice

Entities and attributes

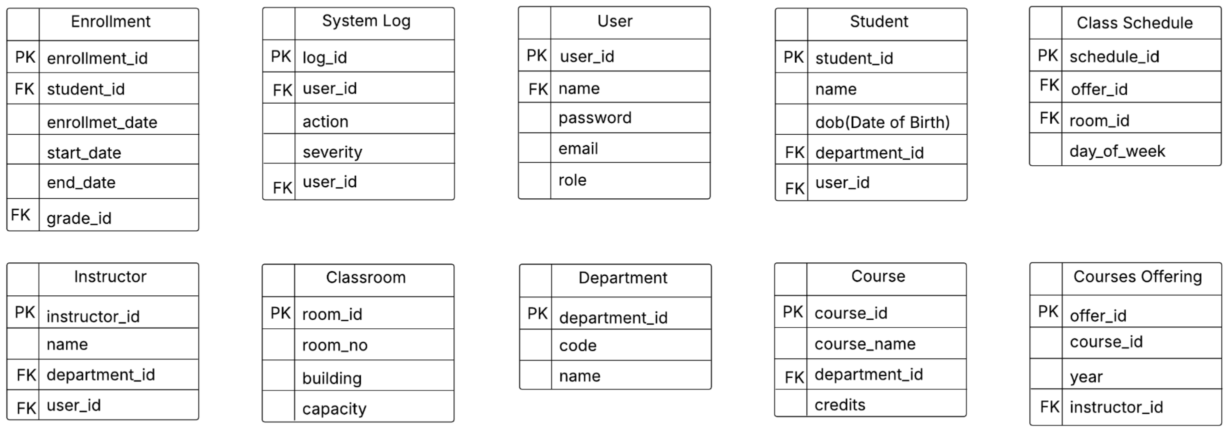


Figure 1.3

Figure 1.3 shows all the entities and attributes.

- PK - Primary Key
- FK - Foreign Key

Relationship

- Department has Course (1:m) → 1 department can associate many courses ,whereas many courses can belong only to one department

- Course has Course Offering (1:m) → 1 course can offer various sub courses related to a particular course and as many course offering is for one course

- Instructor teaches Course Offering (1:m) → 1 instructor can teach many course offering , where as many course offering can be taught by 1 instructor

- Course Offering is scheduled in Class scheduling (m:1) → a course offering has one specific Class Schedule (time/day), but a class schedule can be used for many offerings.

- Class Schedule is held in Classroom (1:m) → one class schedule(time/day) can be scheduled for one or many classrooms Students enroll in Course Offering (m:1) → many students can enroll to 1 course offering of a specific course

- Student enrollment in Enrollment (1:1) → a student can enroll once into enrollment for a specific course.

- Students and Teachers can access System Log using User (user_id),which ensures reliable encryption and safety for the database and the user.

- Course Offering is the main aspect to form a Class Schedule ,Classroom is assigned for Class Schedule.

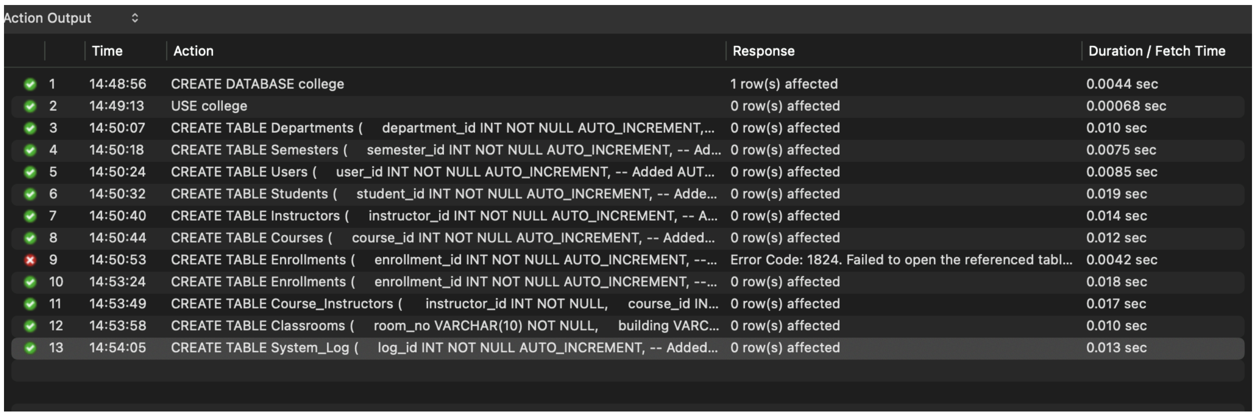
- System log tracks user actions for auditing and security.

SQL Redesigned

```
CREATE DATABASE college;

USE college;

CREATE TABLE Departments (
   department_id INT NOT NULL AUTO_INCREMENT,
   code VARCHAR(10) NOT NULL UNIQUE,
   name VARCHAR(100) NOT NULL,
   PRIMARY KEY (department_id)
);

CREATE TABLE Semesters (
   semester_id INT NOT NULL AUTO_INCREMENT,
   code VARCHAR(10) NOT NULL UNIQUE,
   name VARCHAR(50) NOT NULL,
   start_date DATE,
   end_date DATE,
   PRIMARY KEY (semester_id)
);
```

```
CREATE TABLE Users (
   user_id INT NOT NULL AUTO_INCREMENT,
   username VARCHAR(100) NOT NULL UNIQUE,
   password_hash VARCHAR(255) NOT NULL,
   created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
   PRIMARY KEY (user_id)
);

CREATE TABLE Students (
   student_id INT NOT NULL AUTO_INCREMENT,
   user_id INT,
   given_name VARCHAR(100) NOT NULL,
   family_name VARCHAR(100),
   dob DATE,
   email VARCHAR(255) NOT NULL UNIQUE,
   department_id INT,
   created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
   is_active INT NOT NULL,
   PRIMARY KEY (student_id),
   FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE SET NULL,
   FOREIGN KEY (department_id) REFERENCES Departments(department_id)
);

CREATE TABLE Instructors (
   instructor_id INT NOT NULL AUTO_INCREMENT,
   user_id INT,
   name VARCHAR(200) NOT NULL,
   email VARCHAR(255) NOT NULL UNIQUE,
   department_id INT,
   is_active INT NOT NULL,
   PRIMARY KEY (instructor_id),
   FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE SET NULL,
   FOREIGN KEY (department_id) REFERENCES Departments(department_id)
);

CREATE TABLE Courses (
   course_id INT NOT NULL AUTO_INCREMENT,
   code VARCHAR(20) NOT NULL UNIQUE,
   course_name VARCHAR(200) NOT NULL,
   department_id INT,
   credits INT NOT NULL,
   PRIMARY KEY (course_id),
   FOREIGN KEY (department_id) REFERENCES Departments(department_id),
   CHECK (credits > 0)
);

CREATE TABLE Enrollments (
   enrollment_id INT NOT NULL AUTO_INCREMENT,
   student_id INT NOT NULL,
   course_id INT NOT NULL,
   semester_id INT NOT NULL,
   enrollment_date DATE NOT NULL,
   grade_id INT,
   status VARCHAR(20),
   PRIMARY KEY (enrollment_id),
   UNIQUE (student_id, course_id, semester_id),
   FOREIGN KEY (student_id) REFERENCES Students(student_id) ON DELETE
CASCADE,
   FOREIGN KEY (course_id) REFERENCES Courses(course_id) ON DELETE
RESTRICT,
   FOREIGN KEY (semester_id) REFERENCES Semesters(semester_id) ON DELETE
RESTRICT,
   CHECK (status IN ('enrolled', 'withdrawn', 'completed'))
);

CREATE TABLE Course_Instructors (
   instructor_id INT NOT NULL,
   course_id INT NOT NULL,
   semester VARCHAR(10),
   year INT,
   PRIMARY KEY (instructor_id, course_id, semester, year),
   FOREIGN KEY (instructor_id) REFERENCES Instructors(instructor_id),
   FOREIGN KEY (course_id) REFERENCES Courses(course_id)
);

CREATE TABLE Classrooms (
   room_no VARCHAR(10) NOT NULL,
   building VARCHAR(50),
   capacity INT,
   assigned_course INT,
```

```sql
    PRIMARY KEY (room_no),
    FOREIGN KEY (assigned_course) REFERENCES Courses(course_id)
);

CREATE TABLE System_Log (
    log_id INT NOT NULL AUTO_INCREMENT,
    event_time TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    user_id INT,
    action VARCHAR(100),
    status VARCHAR(20),
    PRIMARY KEY (log_id),
    FOREIGN KEY (user_id) REFERENCES Users(user_id)
);
```



```
Action Output

     Time      Action                                                                           Response                                                Duration / Fetch Time
✓ 1  14:48:56  CREATE DATABASE college                                                          1 row(s) affected                                       0.0044 sec
✓ 2  14:49:13  USE college                                                                      0 row(s) affected                                       0.00068 sec
✓ 3  14:50:07  CREATE TABLE Departments (    department_id INT NOT NULL AUTO_INCREMENT,...     0 row(s) affected                                       0.010 sec
✓ 4  14:50:18  CREATE TABLE Semesters (    semester_id INT NOT NULL AUTO_INCREMENT, -- Ad...   0 row(s) affected                                       0.0075 sec
✓ 5  14:50:24  CREATE TABLE Users (    user_id INT NOT NULL AUTO_INCREMENT, -- Added AUT...    0 row(s) affected                                       0.0085 sec
✓ 6  14:50:32  CREATE TABLE Students (    student_id INT NOT NULL AUTO_INCREMENT, -- Adde...   0 row(s) affected                                       0.019 sec
✓ 7  14:50:40  CREATE TABLE Instructors (    instructor_id INT NOT NULL AUTO_INCREMENT, -- A... 0 row(s) affected                                      0.014 sec
✓ 8  14:50:44  CREATE TABLE Courses (    course_id INT NOT NULL AUTO_INCREMENT, -- Added...    0 row(s) affected                                       0.012 sec
✗ 9  14:50:53  CREATE TABLE Enrollments (    enrollment_id INT NOT NULL AUTO_INCREMENT, --...  Error Code: 1824. Failed to open the referenced tabl...  0.0042 sec
✓ 10 14:53:24  CREATE TABLE Enrollments (    enrollment_id INT NOT NULL AUTO_INCREMENT, --...  0 row(s) affected                                       0.018 sec
✓ 11 14:53:49  CREATE TABLE Course_Instructors (    instructor_id INT NOT NULL,    course_id IN... 0 row(s) affected                                   0.017 sec
✓ 12 14:53:58  CREATE TABLE Classrooms (    room_no VARCHAR(10) NOT NULL,    building VARC...  0 row(s) affected                                       0.010 sec
✓ 13 14:54:05  CREATE TABLE System_Log (    log_id INT NOT NULL AUTO_INCREMENT, -- Added...    0 row(s) affected                                       0.013 sec
```

Design principle in redesigned diagram(schema).

- Redesign choices for the ER diagram were guided by well established database design principles focused on maintainability.performance and integrity.

- Normalisation was applied to achieve at least 3NF(Third Normal Form).Removing redundant data by using reference tables such as departments,semesters and grade codes.This minimises data redundancy and reduces inconsistency risk.

- All relationships between students,courses,instructors,departments and enrollments are enforced using foreign keys,preventing invalid data and ensuring database consistency.

- Indexes on all major join and search columns ensure queries scale efficiently as data grows.

- The User table integrates the role field,while System_Log references users directly,providing robust,auditable trails for every database action.

- The addition of Class Schedule enables mapping of courses according to the time slot allotted to classrooms,which allows flexible,real world scheduling