Tariere Timitimi

Professor Colven Benjamin

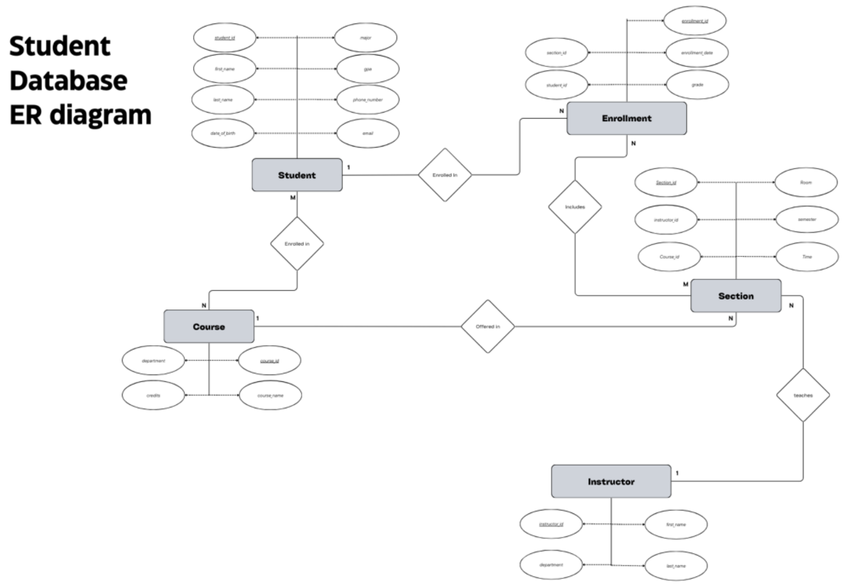
CUS 510 Project #1

20 December 2024

Student Information System Database Schema Design

This report outlines the design and implementation of a database schema for a student information system, which will efficiently store and manage data related to students, courses, sections, instructors, and enrollments. The schema incorporates relational database design principles, normalization, and data integrity constraints to ensure efficient data handling, reduce redundancy, and maintain consistency across the system.

Entity Descriptions



1. Students Table

The **Students** table stores essential information about each student. This includes personal details, academic data, and contact information.

- **student_id** (INT, Primary Key): Unique identifier for each student.
- **first_name** (VARCHAR(50)): First name of the student.
- **last_name** (VARCHAR(50)): Last name of the student.
- **date_of_birth** (DATE): Date of birth.
- **major** (VARCHAR(50)): Academic major.
- **gpa** (DECIMAL(3,2)): Grade Point Average, constrained between 0.00 and 4.00.
- **email** (VARCHAR(100)): Email address.
- **phone_number** (VARCHAR(20)): Contact phone number.
- **address** (TEXT): Residential address.

2. Courses Table

The **Courses** table holds details about the various courses offered by the institution.

- **course_id** (INT, Primary Key): Unique identifier for each course.
- **course_name** (VARCHAR(100)): Name of the course.
- **credits** (INT): Number of credits awarded upon course completion.
- **department** (VARCHAR(50)): Department offering the course.

3. Sections Table

The **Sections** table represents individual course offerings, each associated with a specific instructor, semester, and academic year.

- **section_id** (INT, Primary Key): Unique identifier for each section.
- **course_id** (INT, Foreign Key): References **Courses.course_id**. Defines the course for the section.
- **semester** (VARCHAR(20)): Semester of the course offering (e.g., "Fall", "Spring").
- **Room** (INT): Assigned room for the course
- **instructor_id** (INT, Foreign Key): References **Instructors.instructor_id**. Defines the instructor for the section.
- **Time**(TIME), references the time where section is taught.

4. Instructors Table

The **Instructors** table stores information about the faculty members.

- **instructor_id** (INT, Primary Key): Unique identifier for each instructor.
- **first_name** (VARCHAR(50)): First name of the instructor.
- **last_name** (VARCHAR(50)): Last name of the instructor.
- **department** (VARCHAR(50)): Department to which the instructor belongs.

5. Enrollment Table

The **Enrollment** table tracks which students are enrolled in which course sections.

- **enrollment_id** (INT, Primary Key): Unique identifier for each enrollment record.
- **student_id** (INT, Foreign Key): References **Students.student_id**. Identifies the student enrolled in the section.
- **section_id** (INT, Foreign Key): References **Sections.section_id**. Identifies the section the student is enrolled in.
- **enrollment_date** (DATE): Date the student enrolled.
- **grade** (VARCHAR(2)): Grade received by the student (e.g., "A", "B", "C", "D", "F", "W").

Relationships and Constraints

Entity Relationships

1. **One-to-many relationships:**
    - **Student to Enrollment:** Each student can have multiple enrollments in different sections. A one-to-many relationship exists between the "Student" and "Enrollment" entities.

    - **Course to Section:** One course can have multiple sections. This is represented by a one-to-many relationship between the "Course" and "Section" entities.

    - **Instructor to Section:** One instructor can teach multiple sections. This is represented by a one-to-many relationship between the "Instructor" and "Section" entities.

2. **Many-to-many relationships:**

    - **Course to Student:** Many students can enroll in a course, and a course can have many students enrolled. This many-to-many relationship is facilitated through the "Enrollment" entity. It links the "Student" and "Section" entities, effectively creating a bridge between "Student" and "Course" through their shared association with "Section".
    - **Section to Enrollment:** Multiple students can enroll in a section, and a section can have multiple enrollments. This many-to-many relationship is inherent in the design, as the "Enrollment" entity directly links "Student" and "Section".

Primary Keys

Each table uses a unique primary key to uniquely identify each record:

- **Students**: student_id
- **Courses**: course_id
- **Sections**: section_id
- **Instructors**: instructor_id
- **Enrollment**: enrollment_id

Foreign Keys

- **Sections**: course_id references **Courses.course_id**.
- **Sections**: instructor_id references **Instructors.instructor_id**.
- **Enrollment**: student_id references **Students.student_id**.
- **Enrollment**: section_id references **Sections.section_id**.

Data Integrity Constraints

- **GPA**: The **gpa** attribute is constrained between 0.00 and 4.00 to ensure it falls within the acceptable range.
- **Unique Identifiers**: Attributes such as student_id, instructor_id, course_id, and enrollment_id are unique across their respective tables.
- **Not Null**: Fields like student_id, course_id, and section_id are set to **NOT NULL** to ensure no missing critical data.

Cascading Actions and Update/Delete Policies

To maintain data integrity during updates or deletions, the following cascading actions will be implemented:

- **ON DELETE CASCADE**: When a course is deleted, all associated sections will be deleted automatically.
- **ON DELETE SET NULL**: If an instructor is deleted, their associated sections will have the instructor_id set to NULL, ensuring the sections remain but the instructor field is left unfilled.

Normalization

This schema is designed to adhere to **Third Normal Form (3NF)** to eliminate data redundancy and improve data integrity:

1. **1NF**: All tables have a primary key and contain atomic values.
2. **2NF**: Non-prime attributes are fully dependent on the primary key, and partial dependencies have been eliminated.
3. **3NF**: There are no transitive dependencies, meaning attributes are dependent only on the primary key.

For example, the **Students** table contains no redundant information that could be moved to other tables (such as a separate table for majors or addresses).

Technology Stack

This project will be implemented using **MySQL**. MySQL is a reliable relational database management system (RDBMS) that supports features like:

- Efficient indexing strategies.
- Strong support for foreign keys and cascading actions.
- Data type constraints (e.g., DECIMAL for GPA).
- Mature tools for database design, such as MySQL Workbench.

The use of MySQL would ensure compatibility with industry standards and allows for the easy implementation of the database schema.

Additional Features

1. Prerequisite Tracking

To enforce course prerequisites, a new table **Course_Prerequisites** will be added:

- **course_id** (INT, Foreign Key): References **Courses.course_id**.
- **prerequisite_course_id** (INT, Foreign Key): References **Courses.course_id**.

This will allow us to model prerequisites for each course, ensuring students meet the requirements before enrolling.

2. Automated Grade Calculation

An automated grade calculation system would be developed to compute grades based on students' performance in assignments, exams, and other assessments. This system will integrate with the **Enrollment** table to streamline grading processes and reduce errors.

Handling Missing or Incomplete Data

For non-critical fields like **phone_number** and **address**, NULL values will be allowed. This ensures flexibility in cases where certain data is unavailable or optional. For critical data, like **student_id** or **gpa**, NULL values are not allowed to ensure data integrity.

Naming Consistency

To maintain readability and consistency, all column names are in **snake_case** (e.g., student_id, first_name, gpa). This standardization is crucial for clarity, especially when collaborating with other developers or DBAs.

Conclusion

This database schema provides a robust and flexible design for managing a student information system. The implementation of relational integrity, normalization, cascading actions, and additional features like prerequisite tracking and automated grade calculation will ensure efficient data handling and system performance. With MySQL as the DBMS, this database utilizes a proven platform with strong support for the necessary database operations and constraints.