GROUP 5

22016968 Sarpong Jeffrey Somuah

22042860 Mensah Philemon

22043997 Aidoo-Taylor Kwamena

22231991 Henry Otwey Baidoo

22132121 Amoah Pearl Owusua

22020624 Manal Abdul - Kadi Mohammed

22071976 Eklo Christopher Yao

22049764 Krampah Jonathan

11365437 Captain Godsgift

**Database System for a University Course Registration**

**1. Introduction**

Introduction In modern educational institutions, effective management of course registration is vital to ensure seamless academic operations and student satisfaction. Universities worldwide rely on database systems to handle large volumes of data related to students, professors, courses, and registrations. A well-designed database ensures that all stakeholders have access to accurate and timely information, enhancing administrative efficiency and academic planning.

For example, universities like the Massachusetts Institute of Technology (MIT) and the University of Ghana have adopted robust database systems to manage student registrations, track course offerings, and assign professors. These systems eliminate the inefficiencies of manual processes, such as double-booking of classes, errors in student records, and delays in registration updates. By integrating relational database management systems (RDBMS) like PostgreSQL, universities can store, organize, and retrieve data consistently and securely (Connolly & Begg, 2015).

This report focuses on designing and implementing a database system tailored to a university’s course registration process. The system encompasses four key entities: Students, Professors, Courses, and Registrations, each critical to academic operations. The goal is to ensure data consistency, integrity, and scalability within the database, allowing the university to efficiently manage student enrollment, course assignments, and professor workloads.

Additionally, the report discusses the steps taken to normalize the database to minimize redundancy and ensure optimal performance. Potential challenges, such as handling complex relationships during the translation of an ER diagram into PostgreSQL schema, are also addressed. Ultimately, the proposed database system aims to provide a reliable foundation for academic data management in real-world university settings.

**2. Entity-Relationship (ER) Diagram**

The entities in the ER diagram are Students, Professors, Courses, and Registrations.

A diagram of a company

Description automatically generated

**Entities and Attributes**

* **Students Entity**  
  Contains attributes that store general information about students, including Student\_ID (a unique identifier), Name, Gender, Phone\_No, Email, Department, and Address. Properly structuring student data ensures accuracy in identification and streamlines administrative processes (Elmasri & Navathe, 2016).
* **Professors Entity**  
  Holds similar information, with attributes such as Prof\_ID (a unique identifier), Name, Gender, Phone\_No, Email, and Title, which indicates their academic or administrative position. The inclusion of this entity allows for effective faculty management (Connolly & Begg, 2015).
* **Courses Entity**  
  Provides all relevant details about courses, including Course\_ID (a unique identifier), Name, Code, Prof\_ID (indicating which professor teaches the course), Department, Credit (credit hours), and Description.
* **Registrations Entity**  
  Captures information about student course enrollments with attributes such as Registration\_ID (a unique identifier), Course\_ID, Student\_ID, Semester, and Academic\_Year.

**Relationships Between Entities and Their Cardinalities**

1. **Professors and Courses**
   * A Professor can teach multiple Courses, but each Course is assigned to only one Professor.
   * **Cardinality**: 1:M (One-to-Many).
   * This relationship ensures that courses are uniquely tied to instructors, reflecting real-world teaching assignments (Elmasri & Navathe, 2016).
2. **Courses and Registrations**
   * A Course can have many students registered for it, but each registration corresponds to only one course.
   * **Cardinality**: 1:M (One-to-Many).
   * This relationship prevents duplicate course entries while tracking all registered students (Kroenke & Auer, 2015).
3. **Students and Registrations**
   * A Student can register for multiple courses, and each course registration links a Student to a Course through the Registration entity.
   * **Cardinality**: M:N (Many-to-Many).
   * Many-to-many relationships, when managed effectively, provide flexibility and scalability in database design (Connolly & Begg, 2015).

This database design ensures that all relationships and cardinalities reflect real-world scenarios in a university course registration system. For example, the one-to-many relationship between professors and courses highlights how a professor can teach several courses, while the many-to-many relationship between students and registrations accommodates students enrolling in multiple courses.

**3. Translation of ER Diagram to PostgreSQL Schema**

**Code Breakdown**

**I. Students Table**

The students table contains details about each student. The student\_id serves as the primary key to uniquely identify each student. The gender column restricts values to M or F, ensuring proper data validation. Constraints such as UNIQUE and NOT NULL ensure data consistency and prevent duplication of email or phone numbers.

A black text on a white background

Description automatically generated

**II. Professors Table**

The professors table contains information about faculty members. The professor\_id is the primary key, and constraints such as UNIQUE ensure the integrity of phone numbers and email addresses.

**A black text on a white background

Description automatically generated**

**III. Courses Table**

The courses table includes course-specific data. It enforces a foreign key constraint (professor\_id) to establish a relationship with the professors table. This ensures that every course is linked to a professor.

A white background with black text

Description automatically generated

**IV. Registration Table**

The registration table functions as a linking table to manage the many-to-many relationship between students and courses. It includes foreign keys (student\_id and course\_id) to maintain referential integrity and checks for valid academic years and semesters.

A black text on a white background

Description automatically generated

**4. Key Features Ensuring Consistency and Integrity**

* **Primary Keys**: Each table has a primary key to ensure that rows are uniquely identified.
* **Foreign Keys**: Relationships between tables are enforced through foreign key constraints, which link the registration table to both the students and courses tables, and the courses table to the professors table.
* **Data Validation**: Constraints such as NOT NULL, UNIQUE, and CHECK ensure that invalid or duplicate data cannot be entered. For instance, the CHECK constraints on the gender and semester columns enforce valid values.
* **Referential Integrity**: Foreign key relationships maintain consistency across related tables. For example, deleting a professor or student would fail if they are referenced in other tables unless cascading behavior is defined.

**5. Normalization**

The design of the database for the university's course registration system demonstrates adherence to normalization principles:

* **First Normal Form (1NF)**: Each table ensures atomicity by breaking down data into indivisible units, eliminating any repeating groups or arrays. For instance, student and professor details are stored in separate columns for attributes like name, gender, and phone number.
* **Second Normal Form (2NF)**: All non-key attributes are fully functionally dependent on the primary key. For example, in the courses table, attributes like name, department, and credits depend solely on course\_id.
* **Third Normal Form (3NF)**: The design eliminates transitive dependencies by ensuring non-key attributes depend only on the primary key. For instance, professor\_id links the courses table to the professors table, preventing data redundancy.

By adhering to these normalization steps, the database maintains consistency, integrity, and eliminates redundancy, ensuring efficient data handling and updates.

**6. Issues During the Translation from ER Diagram to PostgreSQL**

**Mapping and Cardinalities**

1. **Professors and Courses (1:M)**
   * The courses table includes a foreign key (professor\_id) referencing the professors table. This ensures that each course is assigned to one professor while allowing each professor to teach multiple courses.
2. **Courses and Registrations (1:M)**
   * The registrations table includes a foreign key (course\_id) referencing the courses table. This ensures that a course can have multiple registration entries while each registration pertains to one course.
3. **Students and Registrations (M:N)**
   * The many-to-many relationship between students and courses is managed through the registrations table, which contains two foreign keys: student\_id (referencing the students table) and course\_id (referencing the courses table).

**Complex Relationships and Attributes**

* Handling composite attributes like a professor's title required simplifying them into atomic fields.
* Multi-valued attributes, such as courses taught in multiple semesters, were managed by adding specific fields (e.g., Semester and Academic\_Year) to relevant tables.

**7. Conclusion**

The proposed database system for university course registration addresses the critical need for efficient academic data management. By leveraging relational database principles and adhering to normalization standards, the system ensures data consistency, integrity, and scalability. The design reflects real-world scenarios, such as the relationships between professors, courses, and students, while mitigating common challenges like redundancy and data integrity issues. Through the use of PostgreSQL, the system provides a robust foundation for seamless course registration processes, supporting the university's academic operations and enhancing overall student and faculty satisfaction. This system exemplifies the integration of technology in modern education, paving the way for more advanced data-driven solutions in the future.

**References**

* Connolly, T. and Begg, C. (2015) *Database Systems: A Practical Approach to Design, Implementation, and Management*. 6th ed. Harlow: Pearson Education.
* Elmasri, R. and Navathe, S.B. (2016) *Fundamentals of Database Systems*. 7th ed. Boston: Pearson.
* Kroenke, D.M. and Auer, D.J. (2015) *Database Concepts*. 7th ed. Boston: Pearson.