**Database Requirements and Decisions**

**1. Introduction**

**Purpose of the Database**

The purpose of this database is to manage student registrations, course details, teacher assignments, billing information, and academic progress, including GPA and degree tracking. It will serve as the central repository for tracking the relationship between students, courses, teachers, and their associated data.

**Scope**

This database will store and manage the following:

* **Student Information**: Including personal details, registration, grades, and billing.
* **Course Information**: Including course details, teacher assignments, and registration details.
* **Teacher Information**: Including contact details and course assignments.
* **Billing Information**: To track tuition payments, billing status, and amounts owed by students.
* **Degree Information**: To track the degrees earned by students, their departments, and credit requirements.

**2. Database Requirements**

**Data Requirements**

The key entities to be stored in the database include:

* **Student**: Information about students (name, contact details, academic records).
* **Course**: Information about the courses offered (course title, department, credits).
* **Teacher**: Information about teachers (name, department, contact details).
* **Registration**: Relationship between students and the courses they are enrolled in.
* **Billing**: Financial transactions, including payments and statuses.
* **Degree**: The degrees students are working toward or have earned.

**Functional Requirements**

* **Registration Management**: Students must be able to register for multiple courses each semester, and the system must track their grades, credits, and progress.
* **Grade Tracking**: Track the grades for students and calculate GPA based on numeric equivalents of letter grades.
* **Degree Tracking**: Keep records of the degrees students are pursuing and whether they have completed the requirements.
* **Billing Management**: Track student tuition payments, manage billing statuses, and generate financial reports.

**Non-Functional Requirements**

* **Performance**: Queries related to student schedules, grades, and billing information must be optimized for speed.
* **Security**: Protect sensitive student data, including personal information and financial records, using encryption and secure access controls.
* **Integrity**: Ensure that data integrity is maintained across all relationships, such as ensuring that students only register for valid courses and that billing records are accurate.

**3. Design Decisions**

**Data Model**

The database is structured using a **relational model** with normalized tables to reduce redundancy and ensure data consistency. The main tables are:

* **Student**: Stores personal and academic details of students.
* **Course**: Stores information about the courses offered.
* **Teacher**: Stores details about the instructors teaching courses.
* **Registration**: A junction table to handle the many-to-many relationship between students and courses.
* **Billing**: Stores tuition and payment records for students.
* **Degree**: Stores details of the degrees pursued by students.

**Key Design Decisions**

* **Normalization**: The database is normalized to **3NF** to ensure there is no redundancy and to maintain data integrity.
* **Primary Keys**: Each table will have a primary key:
  + Student\_Id in **Student**
  + Course\_Id in **Course**
  + Teacher\_Id in **Teacher**
  + Registration\_Id in **Registration**
  + Billing\_Id in **Billing**
  + Degree\_Id in **Degree**
* **Foreign Keys**: Relationships between tables are enforced using foreign keys:
  + Student\_Id in **Registration** references **Student**
  + Course\_Id in **Registration** references **Course**
  + Teacher\_Id in **Course** references **Teacher**
  + Student\_Id in **Billing** references **Student**
  + Degree\_Id in **Degree** references **Degree**

**Indexes**

Indexes will be created to improve query performance, particularly for common join and search operations:

* CREATE INDEX idx\_registration\_student ON Registration(Student\_Id);
* CREATE INDEX idx\_registration\_course ON Registration(Course\_Id);
* CREATE INDEX idx\_billing\_student ON Billing(Student\_Id);
* CREATE INDEX idx\_billing\_status ON Billing(Status);
* CREATE INDEX idx\_degree\_department ON Degree(Department\_Id);
* CREATE INDEX idx\_course\_teacher ON Course(Teacher\_Id);

**Constraints**

* **Not Null Constraints**: Fields that cannot be left empty (e.g., Student\_Name, Course\_Name, Grade).
* **Unique Constraints**: Ensure that certain fields are unique (e.g., Student\_Id and Course\_Id combinations in the **Registration** table).

**4. Relationship Design**

**Entity Relationships**

* **Student and Course**: Many-to-many relationship through the **Registration** table. A student can register for multiple courses, and each course can have many students.
* **Teacher and Course**: One-to-many relationship where each course has a single teacher, but teachers can teach multiple courses.
* **Student and Degree**: Many-to-many relationship, as students can earn multiple degrees over time, and each degree may have different credit requirements.