Ayushi Patel

CSIT355 – Final Project

Course Management System

Database Design Document

This document presents the database design for the Course Management System. The system is intended to support core academic functions such as student enrollment, course scheduling, prerequisite verification, and teaching assignments. This document outlines the relational schema definitions for all entities involved in the system and includes a corresponding Entity-Relationship (ER) diagram.

RELATIONAL SCHEMA DEFINITIONS

1) Students Table

The Students table contains basic information about each student including their unique student ID, name, contact information, and GPA. The GPA is constrained to be within the range of 0 to 4. The student_id serves as the primary key and ensures the uniqueness of each student record.

Schema:

```
Students(
   student_id INTEGER PRIMARY KEY,
   student_name TEXT NOT NULL,
   student_email TEXT UNIQUE NOT NULL,
   student_phone TEXT,
   gpa REAL CHECK (gpa BETWEEN 0 AND 4)
)
```

2) Professors Table

The Professors table stores information about each professor including a unique professor ID, name, department, and contact email. The professor_id acts as the primary key.

Schema:

```
Professors(
```

```
    professor_id INTEGER PRIMARY KEY,
    professor_name TEXT NOT NULL,
    department TEXT NOT NULL,
    professor_email TEXT UNIQUE NOT NULL
)
```

3) Courses Table

The Courses table holds details for each course offered in the system. Each course has a unique course ID, course name, department, description, and a maximum capacity that must not exceed 25 students. The course_id is the primary key. Course names will include at least five courses from the MSU Computer Science department.

Schema:

```
Courses(
    course_id TEXT PRIMARY KEY,
    course_name TEXT NOT NULL,
    credits INTEGER NOT NULL,
    department TEXT NOT NULL,
    description TEXT,
    max_capacity INTEGER CHECK (max_capacity <= 25)
)
```

4) Prerequisites Table

The Prerequisites table is used to represent prerequisite relationships between courses. It contains two foreign keys: course_id and prerequisite_id, both of which reference the Courses table. This structure supports multiple prerequisites per course and ensures all are validated during enrollment.

Schema:

```
Prerequisites(
    course_id TEXT,
    prerequisite_id TEXT,
    PRIMARY KEY(course_id, prerequisite_id),
    FOREIGN KEY(course_id) REFERENCES Courses(course_id),
    FOREIGN KEY(prerequisite_id) REFERENCES Courses(course_id)
)
```

5) Schedule Table

The Schedule table captures the meeting days, times, and classroom locations for courses. Each schedule is linked to a course via a foreign key. Validations ensure that courses are only scheduled on weekdays between 8:00 AM and 10:00 PM. The primary key is a unique schedule_id. Schedule conflicts will be prevented both for students and professors during course enrollment or teaching assignment.

Schema:

```
Schedule(
    schedule_id INTEGER PRIMARY KEY,
    course_id TEXT NOT NULL,
    day_of_week TEXT CHECK(day_of_week IN ('Monday', 'Tuesday', 'Wednesday',
'Thursday', 'Friday')),
    start_time TIME CHECK (start_time >= '08:00' AND start_time <= '22:00'),
    end_time TIME CHECK (end_time > start_time AND end_time <= '22:00'),
    location TEXT,
    FOREIGN KEY(course_id) REFERENCES Courses(course_id)
)
```

6) Teaching Table

The Teaching table tracks the assignment of professors to courses. Each row associates a professor with a course, using a composite primary key. Foreign keys ensure that both the professor and the course exist in their respective tables. Professor teaching history can be derived by querying this table.

Schema:

```
Teaching(
    course_id TEXT,
    professor_id INTEGER,
    PRIMARY KEY(course_id, professor_id),
    FOREIGN KEY(course_id) REFERENCES Courses(course_id),
    FOREIGN KEY(professor_id) REFERENCES Professors(professor_id)
)
```

7) Enrollment Table

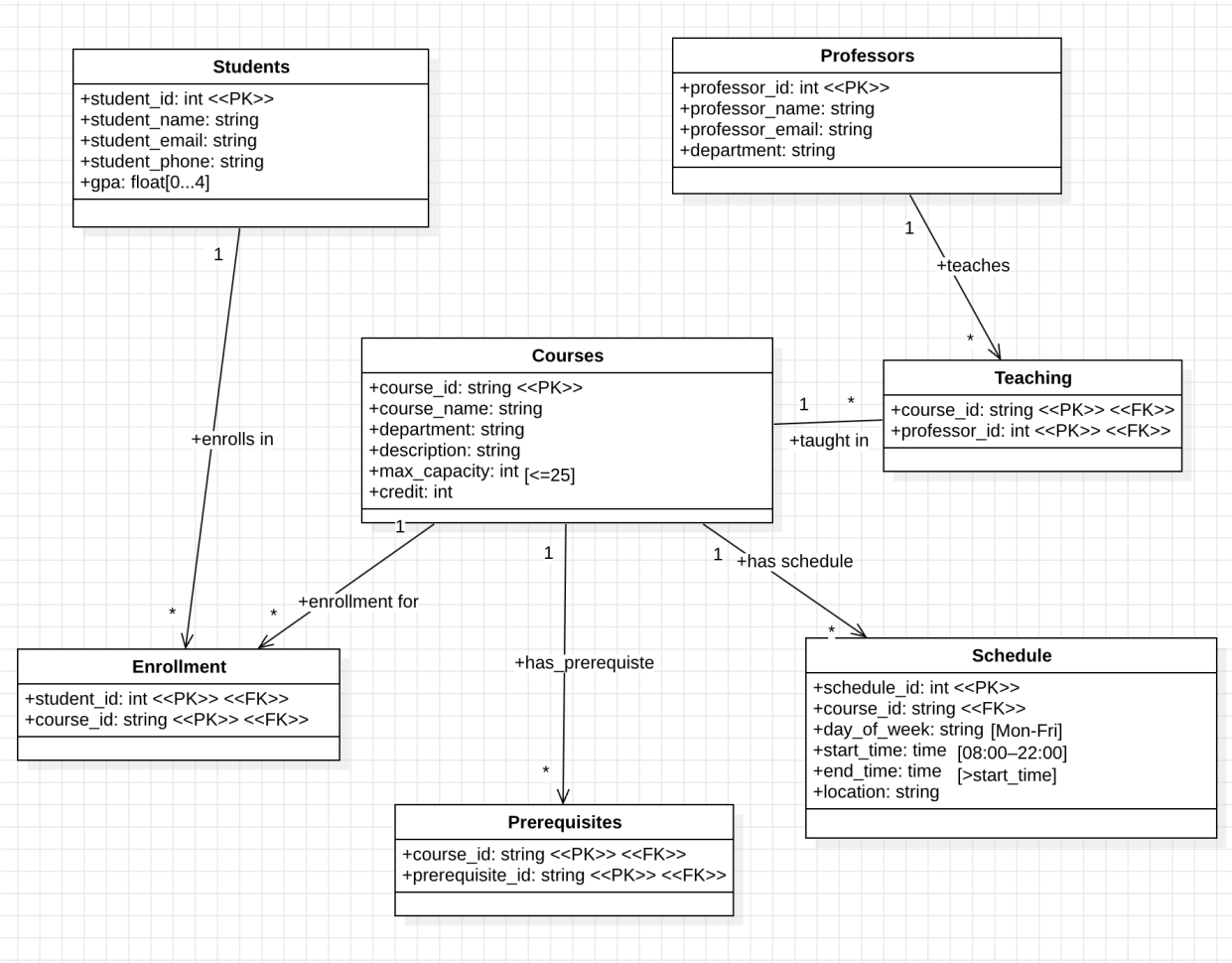
The Enrollment table records the courses in which each student is enrolled. It uses a composite primary key and foreign key constraints to enforce valid student-course relationships. Enrollment logic will check prerequisites, course capacity, schedule conflicts, and prevent duplicate enrollments.

Schema:

```
Enrollment(
   student_id INTEGER,
   course_id TEXT,
   PRIMARY KEY(student_id, course_id),
   FOREIGN KEY(student_id) REFERENCES Students(student_id),
   FOREIGN KEY(course_id) REFERENCES Courses(course_id)
)
```

ENTITY RELATIONSHIP DIAGRAM (ER DIAGRAM)

The following Entity-Relationship (ER) diagram provides a visual representation of the database structure for the Course Management System. It includes the main entities involved in the system, including students, courses, professors, schedules, prerequisites, teaching assignments, and enrollments. The diagram illustrates the relationships between these entities, along with relevant attributes and keys. Cardinalities are indicated to show how entities are connected, and the diagram serves as a blueprint for understanding the logical organization of the system's data.


Students
+student_id: int <<PK>>
+student_name: string
+student_email: string
+student_phone: string
+gpa: float[0...4]
Professors
+professor_id: int <<PK>>
+professor_name: string
+professor_email: string
+department: string
1
+teaches
*
Courses
+course_id: string <<PK>>
+course_name: string
+department: string
+description: string
+max_capacity: int [<=25]
+credit: int
Teaching
+course_id: string <<PK>> <<FK>>
+professor_id: int <<PK>> <<FK>>
1
*
+taught in
1
+enrolls in
*
1
+enrollment for
*
1
+has_prerequiste
*
1
+has schedule
*
Enrollment
+student_id: int <<PK>> <<FK>>
+course_id: string <<PK>> <<FK>>
Prerequisites
+course_id: string <<PK>> <<FK>>
+prerequisite_id: string <<PK>> <<FK>>
Schedule
+schedule_id: int <<PK>>
+course_id: string <<FK>>
+day_of_week: string [Mon-Fri]
+start_time: time [08:00–22:00]
+end_time: time [>start_time]
+location: string