To create a comprehensive system to manage students, courses, grades, and faculty, we will implement the following:

Entity-Relationship (ER) Tables: Tables for students, courses, faculty, and grades.

Stored Procedures: Procedures for enrolling students in courses and updating grades.

Triggers: Automatically update student GPAs when grades change.

SQL Queries: Generate academic transcripts.

1. Database Tables

a. Students Table

This table stores student information.

CREATE TABLE students (

student\_id INT PRIMARY KEY AUTO\_INCREMENT,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

date\_of\_birth DATE,

gpa DECIMAL(3, 2) DEFAULT 0.00

);

b. Courses Table

This table stores details about courses.

CREATE TABLE courses (

course\_id INT PRIMARY KEY AUTO\_INCREMENT,

course\_name VARCHAR(100) NOT NULL,

course\_credits INT NOT NULL

);

c. Faculty Table

This table stores information about faculty members.

CREATE TABLE faculty (

faculty\_id INT PRIMARY KEY AUTO\_INCREMENT,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

department VARCHAR(50)

);

d. Grades Table

This table tracks students enrolled in courses and their grades.

CREATE TABLE grades (

grade\_id INT PRIMARY KEY AUTO\_INCREMENT,

student\_id INT,

course\_id INT,

faculty\_id INT,

grade CHAR(2),

semester VARCHAR(20),

FOREIGN KEY (student\_id) REFERENCES students(student\_id),

FOREIGN KEY (course\_id) REFERENCES courses(course\_id),

FOREIGN KEY (faculty\_id) REFERENCES faculty(faculty\_id)

);

2. Stored Procedures

a. Enroll Students in Courses

This procedure enrolls a student in a course with the assigned faculty member.

DELIMITER $$

CREATE PROCEDURE enroll\_student(

IN student\_id INT,

IN course\_id INT,

IN faculty\_id INT,

IN semester VARCHAR(20)

)

BEGIN

INSERT INTO grades (student\_id, course\_id, faculty\_id, semester)

VALUES (student\_id, course\_id, faculty\_id, semester);

END $$

DELIMITER ;

b. Update Grades

This procedure updates a student's grade for a specific course.

DELIMITER $$

CREATE PROCEDURE update\_grade(

IN grade\_id INT,

IN new\_grade CHAR(2)

)

BEGIN

UPDATE grades

SET grade = new\_grade

WHERE grade\_id = grade\_id;

END $$

DELIMITER ;

3. Trigger: Automatically Update GPA

We calculate the GPA whenever a grade is added or updated. We'll assume a basic grading system where:

A = 4, B = 3, C = 2, D = 1, F = 0.

DELIMITER $$

CREATE TRIGGER update\_student\_gpa

AFTER INSERT OR UPDATE ON grades

FOR EACH ROW

BEGIN

DECLARE total\_credits INT;

DECLARE total\_points DECIMAL(5, 2);

-- Calculate total credits and grade points for the student

SELECT SUM(c.course\_credits), SUM(c.course\_credits \*

CASE NEW.grade

WHEN 'A' THEN 4

WHEN 'B' THEN 3

WHEN 'C' THEN 2

WHEN 'D' THEN 1

ELSE 0

END)

INTO total\_credits, total\_points

FROM grades g

JOIN courses c ON g.course\_id = c.course\_id

WHERE g.student\_id = NEW.student\_id;

-- Update the student's GPA

UPDATE students

SET gpa = total\_points / total\_credits

WHERE student\_id = NEW.student\_id;

END $$

DELIMITER ;

4. SQL Queries

a. Generate Academic Transcripts

This query lists all the courses a student has taken, their grades, and their GPA.

SELECT

s.student\_id,

CONCAT(s.first\_name, ' ', s.last\_name) AS student\_name,

c.course\_name,

g.grade,

c.course\_credits,

g.semester,

s.gpa

FROM

students s

JOIN

grades g ON s.student\_id = g.student\_id

JOIN

courses c ON g.course\_id = c.course\_id

WHERE

s.student\_id = 1 -- Replace with specific student\_id

ORDER BY

g.semester, c.course\_name;

b. List Students in a Course

This query lists all students enrolled in a specific course in a semester.

SELECT

c.course\_name,

CONCAT(s.first\_name, ' ', s.last\_name) AS student\_name,

g.grade,

g.semester

FROM

grades g

JOIN

students s ON g.student\_id = s.student\_id

JOIN

courses c ON g.course\_id = c.course\_id

WHERE

c.course\_id = 1 AND g.semester = 'Fall 2023' -- Replace with specific course\_id and semester

ORDER BY

student\_name;

Example Data

Students Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **student\_id** | **first\_name** | **last\_name** | **date\_of\_birth** | **gpa** |
| 1 | John | Doe | 2000-01-01 | 3.75 |
| 2 | Jane | Smith | 2001-02-15 | 3.50 |

Courses Table

|  |  |  |
| --- | --- | --- |
| **course\_id** | **course\_name** | **course\_credits** |
| 1 | Database Systems | 4 |
| 2 | Algorithms | 3 |

Faculty Table

|  |  |  |  |
| --- | --- | --- | --- |
| **faculty\_id** | **first\_name** | **last\_name** | **department** |
| 1 | Dr. Alice | Brown | Computer Science |

Grades Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **grade\_id** | **student\_id** | **course\_id** | **faculty\_id** | **grade** | **semester** |
| 1 | 1 | 1 | 1 | A | Fall 2023 |
| 2 | 2 | 2 | 1 | B | Fall 2023 |

**Summary:**

Entity Tables: Manage students, courses, faculty, and grades.

Stored Procedures:

Enroll students in courses.

Update grades.

Trigger:

Automatically recalculate and update GPAs when grades change.

SQL Queries:

Generate academic transcripts.

List students enrolled in a course.

This setup ensures a complete, functional database system for managing academic data effectively

**Conclusion:**

The database system for managing students, courses, grades, and faculty provides a robust framework for academic record-keeping and processing. By modeling the entities and their relationships clearly, implementing stored procedures, and incorporating triggers for automated updates, we ensure a highly efficient and dynamic solution. Here’s a summary of the key achievements: