ASSIGNMENT-3

BY P.Venkata Sai Pradeep Reddy[192311364]

Model tables for students, courses, grades, and faculty.

Write stored procedures for enrolling students in courses and updating grades.

Implement triggers to automatically update student GPAs when grades are changed.

Write SQL queries to generate student academic transcripts.

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: