**Project Title: University Management System**

1.

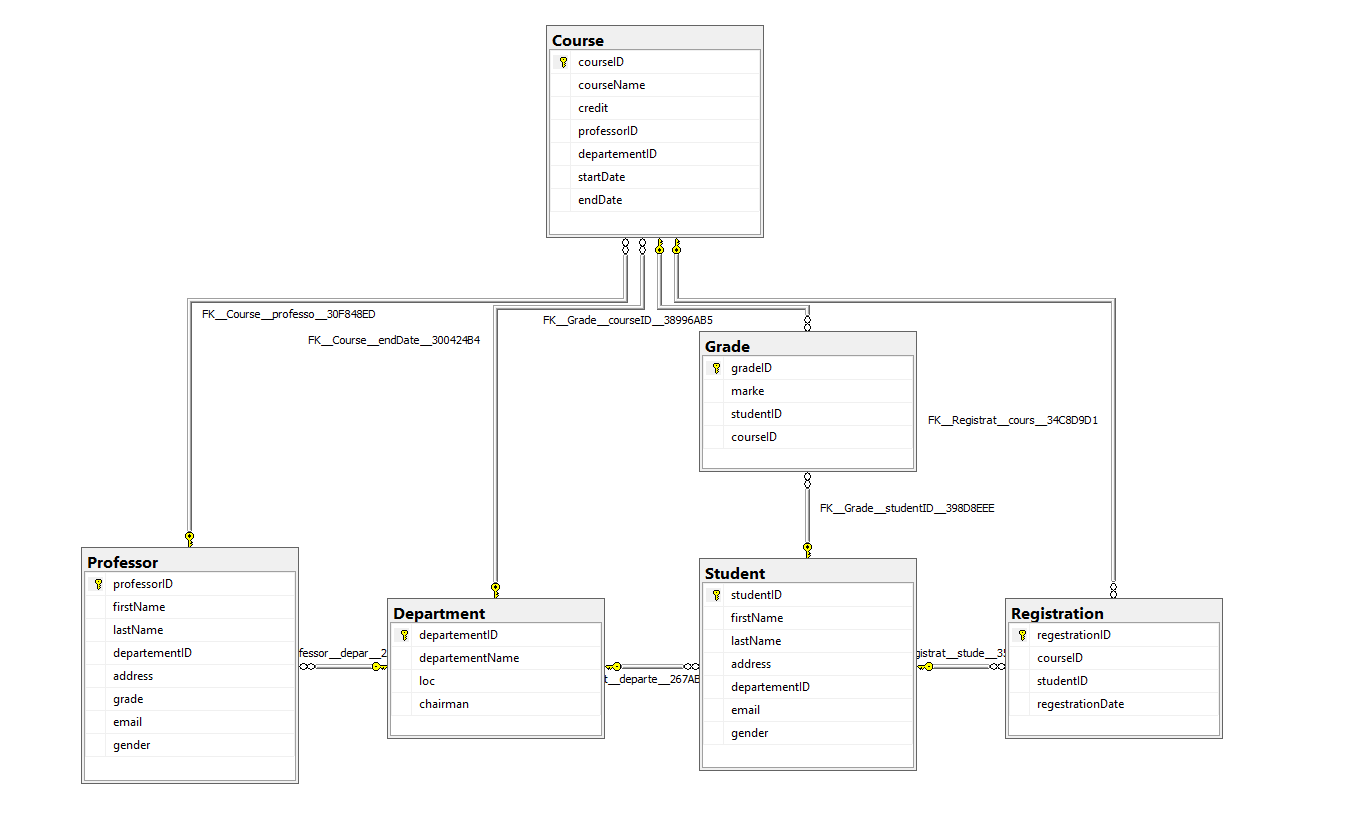
**Department - Student:**

* + Relationship Type: One-to-Many
  + Explanation: One department can have many students, but each student is associated with only one department.
* **Department - Professor:**
  + Relationship Type: One-to-Many
  + Explanation: One department can have many professors, but each professor is associated with only one department.
* **Professor - Course:**
  + Relationship Type: One-to-Many (assuming a professor can teach multiple courses)
  + Explanation: One professor can teach multiple courses, but each course is taught by only one professor.
* **Course - Student:**
  + Relationship Type: Many-to-Many (assuming a student can enroll in multiple courses, and a course can have multiple students)
  + Explanation: A course can have multiple students enrolled, and a student can be enrolled in multiple courses. This is typically implemented using a junction table (e.g., Registration table) that stores pairs of CourseID and StudentID.

registration table is between them

* **Student - Grade:**
  + Relationship Type: One-to-Many (assuming a student can have multiple grades)
  + Explanation: One student can have multiple grades (for different courses), but each grade is associated with only one student.
* **Course - Grade:**
  + Relationship Type: One-to-Many (assuming a course can have multiple grades, one for each student)
  + Explanation: One course can have multiple grades (one for each student who took the course), but each grade is associated with only one course.

2. Database Schema Design:



3. Table Creation (SQL):

CREATE TABLE Department (

departementID INT PRIMARY KEY,

departementName VARCHAR(255),

loc VARCHAR(255),

chairman VARCHAR(255),

)

CREATE TABLE Student(

studentID INT PRIMARY KEY,

firstName VARCHAR(255),

lastName VARCHAR(255),

address VARCHAR(255),

departementID INT,

email VARCHAR(255),

gender VARCHAR(10),

FOREIGN KEY (departementID) REFERENCES Department (departementID)

);

CREATE TABLE Professor(

professorID INT PRIMARY KEY,

firstName VARCHAR(255),

lastName VARCHAR(255),

departementID INT,

address VARCHAR(255),

grade VARCHAR(255),

email VARCHAR(255),

gender VARCHAR(10),

FOREIGN KEY (departementID) REFERENCES Department (departementID)

);

CREATE TABLE Course(

courseID INT PRIMARY KEY,

courseName VARCHAR(255),

credit INT,

professorID INT,

departementID INT,

startDate DATE,

endDate DATE

FOREIGN KEY (departementID) REFERENCES Department (departementID),

FOREIGN KEY (professorID) REFERENCES Professor (professorID)

);

Alter TABLE Department

ADD FOREIGN KEY (chairman) REFERENCES Professor(professorID)

CREATE TABLE Registration (

regestrationID INT PRIMARY KEY,

courseID INT,

studentID INT,

regestrationDate DATE,

FOREIGN KEY (courseID) REFERENCES Course (courseID),

FOREIGN KEY (studentID) REFERENCES Student (studentID));

CREATE TABLE Grade (

gradeID INT PRIMARY KEY,

marke INT,

studentID INT,

courseID INT,

FOREIGN KEY (courseID) REFERENCES Course (courseID),

FOREIGN KEY (studentID) REFERENCES Student (studentID));

4. Data Population:

INSERT INTO Department (departementID, departementName, loc) VALUES

(1, 'Computer Science', 'Building A'),

(2, 'Physics', 'Building B'),

(3, 'Mathematics', 'Building C'),

(4, 'Chemistry', 'Building D'),

(5, 'History', 'Building E'),

(6, 'English', 'Building F'),

(7, 'Biology', 'Building G'),

(8, 'Economics', 'Building H');

INSERT INTO Professor (professorID, firstName, lastName, departementID, address, grade, email, gender) VALUES

(201, 'John', 'Smith', 1, '123 Main St', 'Associate Professor', 'john.smith@example.com', 'Male'),

(202, 'Jane', 'Doe', 2, '456 Oak St', 'Professor', 'jane.doe@example.com', 'Female'),

(203, 'Bob', 'Johnson', 3, '789 Pine St', 'Assistant Professor', 'bob.johnson@example.com', 'Male'),

(204, 'Sara', 'Johnson', 1, '400 Pine St', 'Associate Professor', 'sara.johnson@example.com', 'Female'),

(205, 'Michael', 'Davis', 5, '500 Oak St', 'Professor', 'michael.davis@example.com', 'Male'),

(206, 'Emily', 'White', 1, '600 Elm St', 'Assistant Professor', 'emily.white@example.com', 'Female'),

(207, 'Daniel', 'Adams', 7, '700 Maple St', 'Professor', 'daniel.adams@example.com', 'Male'),

(208, 'Olivia', 'Green', 2, '800 Cedar St', 'Associate Professor', 'olivia.green@example.com', 'Female');

INSERT INTO Student (studentID, firstName, lastName, address, departementID, email, gender) VALUES

(301, 'Alice', 'Williams', '101 Elm St', 1, 'alice.williams@example.com', 'Female'),

(302, 'Charlie', 'Brown', '202 Maple St', 1, 'charlie.brown@example.com', 'Male'),

(303, 'Eva', 'Miller', '303 Cedar St', 2, 'eva.miller@example.com', 'Female'),

(304, 'Frank', 'Johnson', '404 Pine St', 3, 'frank.johnson@example.com', 'Male'),

(305, 'Grace', 'Smith', '505 Oak St', 3, 'grace.smith@example.com', 'Female'),

(306, 'Henry', 'Brown', '606 Elm St', 6, 'henry.brown@example.com', 'Male'),

(307, 'Isabel', 'Davis', '707 Maple St', 7, 'isabel.davis@example.com', 'Female'),

(308, 'Jack', 'Miller', '808 Cedar St', 8, 'jack.miller@example.com', 'Male');

INSERT INTO Course (courseID, courseName, credit, professorID, departementID, startDate, endDate) VALUES

(101, 'Introduction to Programming', 3, 201, 1, '2024-02-01', '2024-05-01'),

(102, 'Classical Mechanics', 4, 202, 2, '2024-03-01', '2024-06-01'),

(103, 'Linear Algebra', 3, 203, 3, '2024-02-15', '2024-05-15'),

(104, 'Organic Chemistry', 4, 204, 4, '2024-03-15', '2024-06-15'),

(105, 'World History', 3, 205, 5, '2024-02-10', '2024-05-10'),

(106, 'English Literature', 3, 206, 6, '2024-02-20', '2024-05-20'),

(107, 'Cell Biology', 4, 207, 7, '2024-03-10', '2024-06-10'),

(108, 'Microeconomics', 3, 208, 8, '2024-02-05', '2024-05-05'),

(109, 'Introduction to Programming', 3, 201, 1, '2024-02-01', '2024-05-01'),

(110, 'Mathemathic', 3, 203, 3, '2024-02-15', '2024-05-15');

INSERT INTO Registration (regestrationID, courseID, studentID, regestrationDate) VALUES

(501, 101, 301, '2024-01-10'),

(502, 101, 302, '2024-01-12'),

(503, 103, 304, '2024-01-15'),

(504, 103 ,305, '2024-02-01'),

(505, 102, 303, '2024-01-20'),

(506, 106, 306, '2024-01-22'),

(507, 107, 307, '2024-01-18'),

(508, 108, 308, '2024-01-15');

INSERT INTO Grade (gradeID, marke, studentID, courseID) VALUES

(601, 90, 301, 101),

(602, 85, 302, 101),

(603, NULL, 301, 103),

(604, 88, 305 , 103),

(605, 92, 303, 102),

(606, 75, 306, 106),

(607, NULL, 307, 107),

(608, 85, 308, 108);

5. Queries and Reports:Create a set of complex SQL queries to retrieve meaningful information from the database, such as:

-- - List all students enrolled in a specific department.

SELECT s.studentID,s.firstName,s.lastName,d.departementName

FROM Student s

INNER JOIN Registration R

ON R.studentID = s.studentID

INNER JOIN Department d

ON s.studentID =r.studentID

--WHERE d.departementName = 'Physics';

--Display the highest and lowest grades for each course.

--We need to group by because we have max and min

SELECT c.courseID,c.courseName,

MAX(marke) AS 'Maximum grade',

MIN (marke) AS 'Minimum grade'

FROM course c

LEFT JOIN grade g

ON c.courseID = g.courseID

GROUP BY c.courseID,c.courseName;

-- Find courses with a high registration count.

--we use having because we hav count and for condition we use having instead of where

SELECT r.courseID,c.courseName,

COUNT (r.studentID) AS 'number\_of\_regestration'

FROM [Registration] r

INNER JOIN Course c

ON c.courseID = r.courseID

group by r.courseID,c.courseName

HAVING

COUNT(r.studentID) = (

SELECT

MAX(registration\_count)

FROM (SELECT COUNT(studentID) AS registration\_count

FROM [Registration] GROUP BY courseID) AS registration\_counts);

6. Normalization: ( The process of separating your data into tables and creating primary keys is called normalization.)

Based on selecting primary keys the DB is normalize

7. Indexing and Performance Optimization:

--FOR GRADE TABLE:

--If you frequently query based on the mark column

CREATE INDEX idx\_mark ON Grade (marke);

--if you often run queries that involve both courseID and studentID together, you might consider creating a composite index on these columns:

CREATE INDEX idx\_course\_student ON Grade (courseID, studentID);

--FOR Registration TABLE:

CREATE INDEX idx\_courseID ON Registration (courseID);

CREATE INDEX idx\_studentID ON Registration (studentID);

CREATE INDEX idx\_course\_student ON Registration (courseID, studentID);

--FOR Course TABLE:

CREATE INDEX idx\_departementID ON Course (departementID);

CREATE INDEX idx\_professorID ON Course (professorID);

CREATE INDEX idx\_departement\_professor ON Course (departementID, professorID);

--FOR Department TABLE:

CREATE INDEX idx\_departementName ON Department (departementName);

--FOR Student TABLE:if you frequently query based on firstName, lastName, or email

CREATE INDEX idx\_firstName ON Student (firstName);

CREATE INDEX idx\_lastName ON Student (lastName);

CREATE INDEX idx\_email ON Student (email);

--FOR Professor TABLE:if you frequently query based on firstName, lastName, or email

CREATE INDEX idx\_firstName\_professor ON Professor (firstName);

CREATE INDEX idx\_lastName\_professor ON Professor (lastName);

CREATE INDEX idx\_email\_professor ON Professor (email);

8. Security Measures:

--creating different user for assigning different role

CREATE LOGIN StudentUser WITH PASSWORD = 'StudentPassword';

CREATE LOGIN ProfessorUser WITH PASSWORD = 'ProfessorPassword';

CREATE LOGIN RegistrationUser WITH PASSWORD = 'RegistrationPassword';

CREATE LOGIN ResponsibleUser WITH PASSWORD = 'ResponsiblePassword';

CREATE USER StudentUser FOR LOGIN StudentUser;

CREATE USER ProfessorUser FOR LOGIN ProfessorUser;

CREATE USER RegistrationUser FOR LOGIN RegistrationUser;

CREATE USER ResponsibleUser FOR LOGIN ResponsibleUser;

--assigning permission based on their need

-- Grant SELECT permission on tables to StudentUser

GRANT SELECT ON OBJECT::dbo.Student TO StudentUser;

GRANT SELECT ON OBJECT::dbo.Registration TO StudentUser;

GRANT SELECT ON OBJECT::dbo.Grade TO StudentUser;

-- Grant SELECT, INSERT, UPDATE, DELETE permissions to ProfessorUser

GRANT SELECT, INSERT, UPDATE, DELETE ON OBJECT::dbo.Course TO ProfessorUser;

GRANT SELECT, INSERT, UPDATE, DELETE ON OBJECT::dbo.Professor TO ProfessorUser;

GRANT SELECT, INSERT, UPDATE, DELETE ON OBJECT::dbo.Grade TO ProfessorUser;

-- Grant permissions to RegistrationUser

GRANT SELECT, INSERT, UPDATE ON OBJECT::dbo.Registration TO RegistrationUser;

-- ResponsibleUser is db owner and he has all permission

ALTER ROLE db\_owner ADD MEMBER ResponsibleUser;

Thank you for your time