

Student Record Management System (SRMS)

Database Project REPort

Sharaiz Ahmed [57288] | Umair Waseem [44296] | Kashan Shahid [53686]

Sir Ihtisham-ullah | Database Management Systems | BSCS 4-1

### Introduction:

The Student Record Management System (SRMS) is a relational database designed to efficiently manage academic records of students. This project aims to eliminate data redundancy, streamline data retrieval, and ensure consistency across records for an educational institution.

### Problem Statement:

Educational institutions face challenges in maintaining accurate and up-to-date records due to manual systems and inconsistencies. SRMS addresses these problems by introducing a normalized, scalable, and secure database system for student records.

### Objectives:

1. Design a fully normalized relational database (up to 3NF)
2. Store comprehensive data related to students, courses, enrollments, and grades
3. Apply constraints like primary keys and foreign keys to ensure data integrity
4. Facilitate efficient and accurate data retrieval and manipulation

### Database Tables and Attributes:

#### STUDENTS Table

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Type** | **Description** |
| StudentID | INT (PK) | Unique student identifier |
| FirstName | VARCHAR | First name |
| LastName | VARCHAR | Last name |
| DateOfBirth | DATE | Date of birth |
| Email | VARCHAR | Email address |
| PhoneNumber | VARCHAR | Contact number |
| Address | TEXT | Residential address |
| Department | VARCHAR | Academic department |
| Semester | VARCHAR | Current semester |

#### COURSES Table

|  |  |  |
| --- | --- | --- |
| Attribute | Type | Description |
| CourseID | INT (PK) | Unique course identifier |
| CourseName | VARCHAR | Name of the course |
| Instructor | VARCHAR | Instructor name |
| CreditHours | INT | Number of credit hours |
| CourseType | VARCHAR | Type of course (Core/Elective) |
| Department | VARCHAR | Offering department |

#### ENROLLMENTS Table

|  |  |  |
| --- | --- | --- |
| Attribute | Type | Description |
| EnrollmentID | INT (PK) | Unique enrollment record |
| StudentID | INT (FK) | Student reference |
| CourseID | INT (FK) | Course reference |
| EnrollmentDate | DATE | Date of enrollment |
| EnrollmentStatus | VARCHAR | Status (Active/Withdrawn) |

#### GRADES Table

|  |  |  |
| --- | --- | --- |
| Attribute | Type | Description |
| GradeID | INT (PK) | Unique grade record |
| StudentID | INT (FK) | Student reference |
| CourseID | INT (FK) | Course reference |
| Grade | CHAR | Letter grade |
| Semester | VARCHAR | Semester name |
| GPA | FLOAT | Grade Point Average |
| Remarks | TEXT | Instructor comments |

### Normalization:

* **1NF:** Each table has a primary key and contains atomic values
* **2NF:** No partial dependency exists on composite keys
* **3NF:** No transitive dependency among non-prime attributes

### ER Diagram Overview:

* One-to-many: One student can enroll in multiple courses
* Many-to-many: Courses can be taken by multiple students (via ENROLLMENTS)
* One-to-one: A grade is specific to one student and one course

### SQL Commands for Table Creation:

CREATE TABLE STUDENTS (

StudentID INT PRIMARY KEY,

FirstName VARCHAR2(50),

LastName VARCHAR2(50),

DateOfBirth DATE,

Email VARCHAR2(100)

);

CREATE TABLE COURSES (

CourseID INT PRIMARY KEY,

CourseName VARCHAR2(100),

Instructor VARCHAR2(100)

);

CREATE TABLE ENROLLMENTS (

EnrollmentID INT PRIMARY KEY,

StudentID INT,

CourseID INT,

EnrollmentDate DATE,

FOREIGN KEY (StudentID) REFERENCES STUDENTS(StudentID),

FOREIGN KEY (CourseID) REFERENCES COURSES(CourseID)

);

CREATE TABLE GRADES (

GradeID INT PRIMARY KEY,

StudentID INT,

CourseID INT,

Grade CHAR(2),

Semester VARCHAR2(20),

FOREIGN KEY (StudentID) REFERENCES STUDENTS(StudentID),

FOREIGN KEY (CourseID) REFERENCES COURSES(CourseID)

);

### SQL Commands for Table Enhancement:

#### Add New Columns:

ALTER TABLE STUDENTS

ADD (

PhoneNumber VARCHAR2(15),

Address VARCHAR2(255),

Department VARCHAR2(100),

Semester VARCHAR2(20)

);

ALTER TABLE COURSES

ADD (

CreditHours NUMBER(2),

CourseType VARCHAR2(50),

Department VARCHAR2(100)

);

ALTER TABLE ENROLLMENTS

ADD (

EnrollmentStatus VARCHAR2(20)

);

ALTER TABLE GRADES

ADD (

GPA NUMBER(3,2),

Remarks VARCHAR2(255)

);

#### Insert Sample Data:

INSERT INTO STUDENTS (StudentID, FirstName, LastName, DateOfBirth, Email, PhoneNumber, Address, Department, Semester)

VALUES

(57288, 'Sharaiz', 'Romee', TO\_DATE('1993-09-05', 'YYYY-MM-DD'), 'sharaiz.romee@gmail.com', '03001234567', 'Lahore, Pakistan', 'Computer Science', '4th'),

(1, 'Ali', 'Khan', TO\_DATE('2001-12-05', 'YYYY-MM-DD'), 'ali.khan@example.com', '03111234567', 'Islamabad, Pakistan', 'Computer Science', '4th'),

(2, 'Sara', 'Ahmed', TO\_DATE('2002-09-23', 'YYYY-MM-DD'), 'sara.ahmed@example.com', '03211234567', 'Karachi, Pakistan', 'Computer Science', '4th');

INSERT INTO COURSES (CourseID, CourseName, Instructor, CreditHours, CourseType, Department)

VALUES

(101, 'Database Systems', 'Mr. Ihtisham-Ullah', 3, 'Core', 'Computer Science'),

(102, 'Computer Architecture', 'Dr. Raza', 3, 'Core', 'Computer Science');

INSERT INTO ENROLLMENTS (EnrollmentID, StudentID, CourseID, EnrollmentDate, EnrollmentStatus)

VALUES

(1, 57288, 101, TO\_DATE('2024-09-01', 'YYYY-MM-DD'), 'Active'),

(2, 57288, 102, TO\_DATE('2024-09-01', 'YYYY-MM-DD'), 'Active'),

(3, 1, 101, TO\_DATE('2024-09-01', 'YYYY-MM-DD'), 'Active'),

(4, 1, 102, TO\_DATE('2024-09-01', 'YYYY-MM-DD'), 'Active'),

(5, 2, 101, TO\_DATE('2024-09-01', 'YYYY-MM-DD'), 'Active'),

(6, 2, 102, TO\_DATE('2024-09-01', 'YYYY-MM-DD'), 'Active');

INSERT INTO GRADES (GradeID, StudentID, CourseID, Grade, Semester, GPA, Remarks)

VALUES

(1, 57288, 101, 'A', 'Fall 2024', 4.0, 'Excellent'),

(2, 57288, 102, 'A-', 'Fall 2024', 3.7, 'Very Good'),

(3, 1, 101, 'B+', 'Fall 2024', 3.3, 'Good performance'),

(4, 1, 102, 'A-', 'Fall 2024', 3.7, 'Very Good'),

(5, 2, 101, 'A', 'Fall 2024', 4.0, 'Excellent'),

(6, 2, 102, 'B+', 'Fall 2024', 3.3, 'Good performance');

### Conclusion:

The SRMS project provides a normalized, well-structured, and scalable solution for managing student academic records. It improves data integrity, enhances query efficiency, and supports institutional data management needs effectively.

By carefully designing the database schema to follow 3NF normalization, we have minimized redundancy and ensured consistency across all entities. This ensures that updates to records can be made reliably without introducing inconsistencies.

Furthermore, with the use of foreign key relationships and clear entity relationships, our system guarantees that referential integrity is maintained at all times. The added attributes such as GPA, Remarks, Phone Number, and Address bring the schema closer to real-world practical scenarios that institutions require for detailed student management.

From an academic perspective, this project has allowed the team to understand and apply fundamental database principles, such as relational modeling, SQL operations, ER diagramming, and normalization. Technically, the implementation on Oracle DBMS ensures compatibility with enterprise-level systems, offering robustness and scalability.

Moving forward, this system can be expanded to support additional modules like attendance tracking, transcript generation, and academic progress analysis. It can also be integrated with front-end applications to provide an interactive user experience for students and administrators.