

COMPUTER SCIENCE DEPARTMENT

Total Marks: _____

Obtained Marks: _____

LAB: Database System

Project Report: University Management System

Submitted To: Mam Kashia Riaz

Date of Submission: 25-October-2024

Project Members

S.NO	Student Name	Reg.no
01	Ubaid Bin Waris	2212416
02	Mushahid Hussain	2212408
03	Muhammad Salman	2212400
04	Jehanzeb Khalid	2212391



Table of Contents

Type chapter title (level 1)	1
Type chapter title (level 2).....	2
Type chapter title (level 3).....	3
Type chapter title (level 1)	4
Type chapter title (level 2).....	5
Type chapter title (level 3).....	6

Introduction

The University Database System is designed to manage and streamline essential academic data within a university setting. This system provides an organized structure for storing and managing data related to classrooms, courses, faculty, and schedules. It is built on a relational database schema, ensuring data integrity, efficient retrieval, and logical organization through structured tables and relationships. The project includes creating, inserting, updating, and deleting data entries across various tables, each of which represents a key element of the university's administrative processes. This system is designed to improve data accessibility, reduce redundancy, and support administrative decision-making.

Specific Features

- **Classroom Management**
The database includes a classrooms table, which stores details about each room available for academic and administrative purposes. Each room has a unique `room_id` and may also have attributes like `room_name` and `capacity`. This feature ensures that every classroom's usage can be effectively scheduled and tracked.
- **Course Management**
The courses table captures essential information for all courses offered at the university, such as `course_id` and `course_name`. This table allows for straightforward querying of course details, simplifying course enrollment and scheduling operations.
- **Faculty Information Management**
A faculty table is included to store records of faculty members. Key attributes might include `faculty_id`, `faculty_name`, and `department`. By organizing faculty data, the system makes it easier to assign faculty to courses and manage teaching schedules.
- **Schedule and Timetable Management**
The schedules table plays a critical role in organizing when and where classes are conducted. It includes fields for `schedule_id`, `course_id`, `room_id`, `faculty_id`, `day`, `start_time`, and `end_time`, all linked through foreign key relationships. This table is essential for managing time allocations, reducing scheduling conflicts, and ensuring classrooms and faculty are appropriately assigned.
- **Data Integrity through Relationships**
The system uses foreign key constraints to enforce relationships between tables, such as linking `room_id` in the schedules table to the classrooms table. This setup ensures that data remains consistent and prevents the use of non-existent references, enhancing overall data integrity.
- **CRUD Operations**
The database supports all CRUD (Create, Read, Update, Delete) operations, allowing administrators to efficiently manage data. For instance:
Create: Adding new courses, classrooms, and faculty records.
Read: Retrieving information about courses, schedules, or classroom assignments.
Update: Modifying schedules or reallocating rooms if there are scheduling conflicts.
Delete: Removing outdated or unnecessary entries, like old schedules or courses no longer offered.

Data Manipulation: The database structure allows for efficient data manipulation, making it possible to generate reports on classroom usage, faculty workloads, and course schedules. For example, queries can determine which classrooms are used most frequently or analyze faculty teaching hours.

- **User-Friendly Access for Database Administrators**

The database design simplifies complex queries by using a well-organized schema, allowing administrators to retrieve, update, or report data quickly. For instance, finding all courses taught by a specific faculty member or listing available classrooms at a given time can be done through single queries.

These features illustrate the system's potential to provide a comprehensive data management solution for university administrators. You can further elaborate on each feature by explaining the specific tables and attributes involved, as well as the SQL commands used to perform actions within each feature.

Database Schema

The database schema for your University Database System would outline the structure of each table, including primary keys, foreign keys, and relationships between tables. Here's an example schema based on the features

1. Departments Table

Stores information about departments within the university, including a unique department ID and the head of the department.

```
create table Departments (  
    department_id int primary key auto_increment,  
    department_name varchar(100) not null,  
    head_faculty_id int unique  
);
```

2. Students Table

Contains information about students, including personal details and the department they're enrolled in.

```
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,  
    email varchar(100) unique not null,  
    phone_number varchar(15),  
    enrollment_date date,  
    department_id int,  
    FOREIGN KEY (department_id) REFERENCES Departments(department_id)  
);
```

3. Faculty Table

Contains information about faculty members, including personal details and the department they are associated with.

```
create table Faculty (  
    faculty_id int primary key auto_increment,  
    first_name varchar(50) not null,  
    last_name varchar(50) not null,  
    email varchar(100) unique not null,  
    phone_number varchar(15),  
    hire_date date,  
    department_id int,  
    FOREIGN KEY (department_id) REFERENCES Departments(department_id),  
    unique (department_id, faculty_id)  
);
```

4. Courses Table

Stores details about courses, including the department offering the course.

```
create table Courses (  
    course_id int primary key auto_increment,  
    course_name varchar(100) not null,  
    credits int,  
    department_id int,  
    FOREIGN KEY (department_id) REFERENCES Departments(department_id)  
);
```

5. Enrollments Table

Tracks course enrollments for students, including grades. This is a weak entity dependent on both Students and Courses.

```
create table Enrollments (  
    enrollment_id int primary key auto_increment,  
    student_id int,  
    course_id int,  
    enrollment_date date,  
    grade varchar(2),  
    FOREIGN KEY (student_id) REFERENCES Students(student_id),  
    FOREIGN KEY (course_id) REFERENCES Courses(course_id),  
    UNIQUE (student_id, course_id)  
);
```

6. Classrooms Table

Manages classrooms, including unique room numbers and attributes like building and capacity.

```
create table Classrooms (
    room_id int primary key auto_increment,
    room_number varchar(10) unique not null,
    building varchar(50),
    capacity int
);
```

7. Schedules Table

Schedules classes, linking them to courses, classrooms, and faculty members. This is a weak entity dependent on both Courses and Faculty.

```
create table Schedules (
    schedule_id int primary key auto_increment,
    course_id int,
    room_id int,
    faculty_id int,
    day ENUM('Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'),
    time_slot time,
    FOREIGN KEY (course_id) REFERENCES Courses(course_id),
    FOREIGN KEY (room_id) REFERENCES Classrooms(room_id),
    FOREIGN KEY (faculty_id) REFERENCES Faculty(faculty_id),
    UNIQUE (course_id, room_id, day, time_slot)
);
```

8. Fees Table

Tracks fees for each student, with status indicating whether fees are paid or unpaid.

```
create table Fees (
    fee_id int primary key auto_increment,
    student_id int,
    amount decimal(10, 2),
    due_date date,
    status ENUM('Paid', 'Unpaid'),
    FOREIGN KEY (student_id) REFERENCES Students(student_id)
);
```

9. Library Table

Manages books in the university's library, including loaned books and their status.

```
create table Library (
    book_id int primary key auto_increment,
    title varchar(100) not null,
    author varchar(100),
    status ENUM('Available', 'Loaned'),
    isbn varchar(20) unique,
    loaned_to_student_id int,
    FOREIGN KEY (loaned_to_student_id) REFERENCES Students(student_id)
);
```

Schema Relationships and Constraints Summary

- **Primary Keys** Each table has a primary key (department_id, student_id, faculty_id, course_id, etc.) to uniquely identify records.
- **Foreign Keys** Foreign key constraints ensure that relationships between tables (like department_id in Faculty and Students) maintain data integrity.
- **Unique Constraints** Unique constraints, such as on email in Students and Faculty, ensure that certain attributes remain distinct across records.
- **Enumerated Data Types:** For attributes with limited values (e.g., status in Library and Fees tables), ENUM data types improve data accuracy and constraints.

This schema provides a comprehensive structure for managing university-related data, from departments and faculty to courses, classrooms, schedules, fees, and library books. Each table is organized to support the logical relationships between different entities within the university system.

Data Manipulation: Insert, Update, Delete, and Select Operations

In the University Database System, data manipulation includes inserting new records, updating existing ones, deleting as necessary, and selecting data for display or reporting. Below are the commands and examples for each of these operations:

➤ Inserting Data

To populate tables with initial data, INSERT INTO commands are used. Each table requires specific values based on its attributes and foreign key dependencies.

```
INSERT INTO Departments (department_id, department_name, head_faculty_id) VALUES
(1, 'Computer Science', 1),
(2, 'Business Administration', 2),
(3, 'Mathematics', 3),
(4, 'Physics', 4),
(5, 'Chemistry', 5),
(6, 'Electrical Engineering', 6),
(7, 'Mechanical Engineering', 7),
(8, 'Civil Engineering', 8),
(9, 'Psychology', 9),
(10, 'Economics', 10);
```

➤ Updating Data

The UPDATE command allows modifying existing data in the tables, commonly used for updating contact details or course assignments.

```
UPDATE Departments
SET department_name = 'Geography'
WHERE department_name = 'Economics';
```

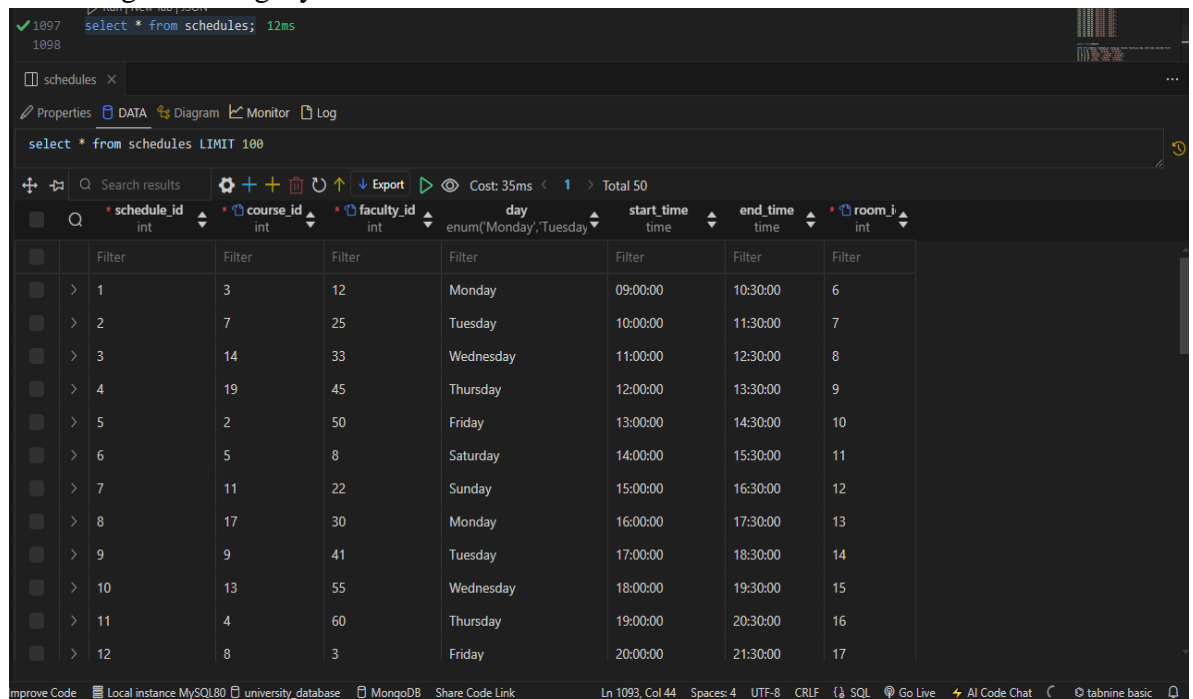
➤ Deleting Data

To remove records from a table, use the DELETE command. When deleting data from tables with foreign key dependencies, ensure that constraints are met or use cascading deletions where appropriate.

```
DELETE FROM enrollments
WHERE student_id = 52; 20ms AffectedRows: 3
```

➤ Selecting Data for Display

The SELECT command retrieves data from tables, often with JOINS to connect related tables. This is used for displaying information, generating reports, and validating data integrity.



schedule_id	course_id	faculty_id	day	start_time	end_time	room_id
1	3	12	Monday	09:00:00	10:30:00	6
2	7	25	Tuesday	10:00:00	11:30:00	7
3	14	33	Wednesday	11:00:00	12:30:00	8
4	19	45	Thursday	12:00:00	13:30:00	9
5	2	50	Friday	13:00:00	14:30:00	10
6	5	8	Saturday	14:00:00	15:30:00	11
7	11	22	Sunday	15:00:00	16:30:00	12
8	17	30	Monday	16:00:00	17:30:00	13
9	9	41	Tuesday	17:00:00	18:30:00	14
10	13	55	Wednesday	18:00:00	19:30:00	15
11	4	60	Thursday	19:00:00	20:30:00	16
12	8	3	Friday	20:00:00	21:30:00	17

These operations form the backbone of CRUD (Create, Read, Update, Delete) functionality, enabling the system to manage university data effectively and maintain consistency.

Conclusion

The University Database System effectively organizes and manages essential university data, including departments, students, faculty, courses, enrollments, and resources like classrooms and the library. By implementing a well-structured schema with clear relationships and constraints, the system maintains data integrity, facilitates efficient data retrieval, and supports a wide range of operations. Through the use of SQL commands for inserting, updating, deleting, and querying data, the system can handle typical university functions with accuracy and ease.

This project not only demonstrates fundamental database principles but also showcases the practical use of SQL in real-world applications. By managing relationships between different entities, enforcing constraints, and maintaining referential integrity, the system is designed to be scalable, reliable, and secure. This foundation will support future expansions, including more complex features and reporting tools, ultimately making it a valuable asset for university administration and academic management.