Database Systems Lab

Project Proposal

Housing Society System

Submitted by:

M AYAN SAJID	2024-CS-661
M SULTAN AWAIS	2024-CS-667
LOZINA IMSAL	2024-CS-666

Submitted to:

MS. RIDA

Dated: 28th April, 2025.

Department of Computer Science

University of Engineering and Technology Lahore, New Campus.

1. Introduction

A **Housing Society Management System** requires a robust and well-structured **database system** to manage residents, financial transactions, maintenance, security, and complaints. This proposal outlines the development and implementation of a **relational database system** using **SQL (Structured Query Language)** to manage all aspects of housing society management efficiently.

2. Objectives

The main objectives of using SQL for this system include:

- **Data storage and retrieval:** Efficient **data storage and retrieval** for residents, properties, and transactions.
- **Data integrity and Security:** Ensuring **data integrity and security** with constraints and access control.
- **Security Issues:** Managing **security logs, visitor entries, and complaint tracking**.
- **Querying and Reporting**: Providing structured **querying and reporting** capabilities.

3. System Requirements

3.1 Functional Requirements

- **Resident Management**: Storing owner and tenant details.
- **Property Management**: Tracking apartments, ownership, and occupancy status.
- **Financial Transactions**: Recording maintenance payments, utility bills, and dues.
- **Complaint Management**: Logging and tracking complaint resolutions.
- **Security & Visitor Management**: Managing visitor logs, security records, and access control.
- **Normalization:** Normalize the database to minimize redundancy and ensure efficient data storage.

3.2 Non-Functional Requirements

- **Data Integrity**: Using SQL constraints like `PRIMARY KEY`, `FOREIGN KEY`, `UNIQUE`, and `CHECK` to enforce accuracy.
- **Security**: Implementing user-based access control with SQL permissions.
- **Performance Optimization**: Using indexing, stored procedures, and views for efficient querying.
- **Scalability**: Designing a **normalized database structure** that supports multiple housing societies.

Challenges and Strategies:

Anticipated challenges include ensuring data consistency, optimizing performance, and meeting legal compliance requirements. Strategies for overcoming these challenges include thorough testing, performance tuning, and regular audits to ensure compliance.

Conceptual Model:

The database's conceptual model is represented using Entity-Relationship Diagrams (ERDs), illustrating relationships between entities. These relationships help visualize the connections between different entities and ensure a comprehensive database design.

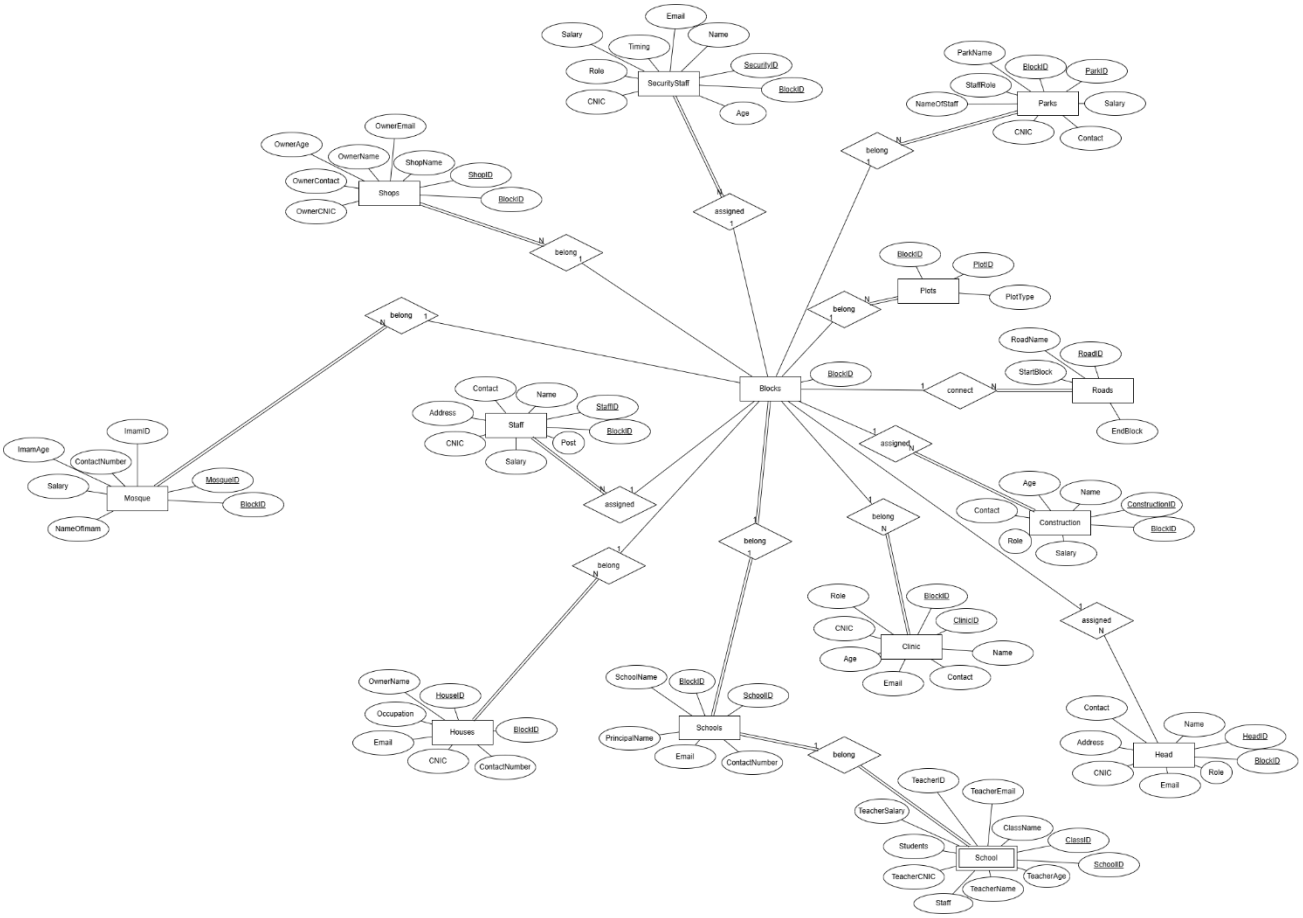

Figure 1. ERD

4. Database Design Using SQL

4.1 Querying & Reporting

SQL will be used for:

- Retrieving **resident details** based on apartment numbers.
- Generating **monthly financial reports** for maintenance and utility payments.
- Tracking **pending complaints and response times**.
- Managing **visitor logs and security alerts**.

5. Implementation Plan Using SQL

- **Database Design**: Creating an optimized schema with relationships.
- **SQL Query Development**: Writing queries for CRUD operations.
- **Normalization:** Normalize the database to minimize redundancy and ensure efficient data storage.
- **Stored Procedures & Triggers**: Automating tasks such as payment reminders.
- **Indexing & Optimization**: Enhancing performance with indexing strategies.
- **Testing & Deployment**: Ensuring security, accuracy, and reliability.

6. Benefits of Using SQL

- **Data Consistency**: Enforces integrity using constraints.
- **Security**: Implements role-based access control (`GRANT` & `REVOKE`).
- **Scalability**: Efficiently manages large volumes of data.
- **Flexibility**: Allows complex queries and reporting.
- **Performance**: Optimized indexing and stored procedures for fast access.

Tables Details

➢ Staff

ATTRIBUTES

- Staff-ID INT PRIMARY KEY IDENTITY (1,1)
- Name NVARCHAR (100) NOT NULL
- Contact NVARCHAR (15)
- Address NVARCHAR (255)
- Salary DECIMAL (10, 2)
- CNIC NVARCHAR (15) UNIQUE

- Post NVARCHAR (50) CHECK (Post IN ('Assistant Manager', 'Sweeper', 'Dealer', 'Electrician', 'General Staff'))
- Block-ID CHAR (1) FOREIGN KEY REFERENCES Blocks (Block-ID)

➢ Construction

ATTRIBUTES

- Construction-ID INT PRIMARY KEY IDENTITY (1,1)
- Name NVARCHAR (100) NOT NULL
- Age INT
- Salary DECIMAL (10, 2)
- Contact NVARCHAR (15)
- Role NVARCHAR (50) CHECK (Role IN ('Attender', 'Laborer', 'Helper'))
- Block-ID CHAR (1) FOREIGN KEY REFERENCES Blocks (Block-ID)

➢ Head

ATTRIBUTES

- Head-ID INT PRIMARY KEY IDENTITY (1,1)
- Name NVARCHAR (100) NOT NULL
- Email NVARCHAR (100)
- Contact NVARCHAR (15)
- CNIC NVARCHAR (15) UNIQUE
- Address NVARCHAR (255)
- Role NVARCHAR (50) CHECK (Role IN ('Owner', 'Manager'))
- Block-ID CHAR (1) FOREIGN KEY REFERENCES Blocks (Block-ID)

➢ School

ATTRIBUTES

- Class-ID INT PRIMARY KEY IDENTITY (1,1)
- Class-Name NVARCHAR (50) NOT NULL
- Teacher-ID INT
- Students INT
- Staff INT
- Teacher-Name NVARCHAR (100)
- Teacher-Age INT
- Teacher-CNIC NVARCHAR (15) UNIQUE
- Teacher-Salary DECIMAL (10, 2)
- Teacher-Email NVARCHAR (100)
- School-ID INT FOREIGN KEY REFERENCES Schools (School-ID)

➢ Security-Staff

ATTRIBUTES

- Security-ID INT PRIMARY KEY IDENTITY (1,1)
- Name NVARCHAR (100) NOT NULL
- Age INT
- CNIC NVARCHAR-(15) UNIQUE
- Email NVARCHAR-(100)
- Salary DECIMAL-(10, 2)
- Timing NVARCHAR (50)
- Role NVARCHAR (50) CHECK (Role IN ('Head', 'Guard'))
- Block-ID CHAR (1) FOREIGN KEY REFERENCES Blocks (Block-ID)

➢ Shops

ATTRIBUTES

- Shop-ID INT PRIMARY KEY IDENTITY (1,1)
- Shop-Name NVARCHAR (100) NOT NULL
- Owner-Name NVARCHAR (100) NOT NULL
- Owner-Age INT
- Owner-Contact NVARCHAR (15)
- Owner-CNIC NVARCHAR (15) UNIQUE
- Owner-Email NVARCHAR (100)
- Block-ID CHAR (1) FOREIGN KEY REFERENCES Blocks (Block-ID)

➢ Blocks

ATTRIBUTES

- Block-ID CHAR (1) PRIMARY KEY CHECK (Block-ID IN ('A', 'B', 'C', 'D'))

➢ Mosque

ATTRIBUTES

- Mosque-ID INT PRIMARY KEY IDENTITY (1,1)
- Name-Of-Imam NVARCHAR (100) NOT NULL
- Imam-ID NVARCHAR (15) UNIQUE
- Imam-Age INT
- Salary DECIMAL (10, 2)
- Contact-Number NVARCHAR (15)
- Block-ID CHAR (1) FOREIGN KEY REFERENCES Blocks (Block-ID)

➢ Plots

ATTRIBUTES

- Plot-ID INT PRIMARY KEY IDENTITY (1,1)
- Plot-Type NVARCHAR (50) CHECK (Plot-Type IN ('Commercial', 'Residential'))
- Block-ID CHAR (1) FOREIGN KEY REFERENCES Blocks (Block-ID)

➢ Clinic

ATTRIBUTES

- Clinic-ID INT PRIMARY KEY IDENTITY (1,1)
- Name NVARCHAR (100) NOT NULL
- Contact NVARCHAR (15)
- Email NVARCHAR (100)
- Age INT
- CNIC NVARCHAR (15) UNIQUE
- Role NVARCHAR (50) CHECK (Role IN ('Doctor', 'Pharmacist', 'Receptionist'))
- Block-ID CHAR (1) FOREIGN KEY REFERENCEs Blocks (Block-ID)

➢ Houses

ATTRIBUTES

- House-ID INT PRIMARY KEY IDENTITY (1,1)
- Owner-Name NVARCHAR (100) NOT NULL
- Occupation NVARCHAR (100)
- CNIC NVARCHAR (15) UNIQUE
- Email NVARCHAR (100)
- Contact-Number NVARCHAR (15)
- Block-ID CHAR (1) FOREIGN KEY REFERENCES Blocks (Block-ID)

➢ Roads

ATTRIBUTES

- Road-ID INT PRIMARY KEY IDENTITY (1,1)
- Road-Name NVARCHAR (50) NOT NULL
- Start-Block CHAR (1) FOREIGN KEY REFERENCES Blocks (Block-ID)
- End-Block CHAR (1) FOREIGN KEY REFERENCES Blocks (Block-ID)

➢ Parks

ATTRIBUTES

- Park-ID INT PRIMARY KEY IDENTITY (1,1)

- Park-Name NVARCHAR (100) NOT NULL
- Staff-Role NVARCHAR (50)
- Name-Of-Staff NVARCHAR (100)
- CNIC NVARCHAR (15) UNIQUE
- Contact NVARCHAR (15)
- Salary DECIMAL (10, 2)
- Block-ID CHAR (1) FOREIGN KEY REFERENCES Blocks (Block-ID)

➢ Schools

ATTRIBUTES

- School-ID INT PRIMARY KEY IDENTITY (1,1)
- School-Name NVARCHAR (100) NOT NULL
- Principal-Name NVARCHAR (100)
- Contact-Number NVARCHAR (15)
- Email NVARCHAR (100)
- Block-ID CHAR (1) UNIQUE FOREIGN KEY REFERENCES Blocks (Block-ID)

Relationships between Entities

- Many houses belong to one block.
- Many shops belong to one block.
- Many security staff members are assigned to one block.
- Many staff members are assigned to one block.
- Many construction projects are assigned to one block.
- Many mosques belong to one block.
- Many plots belong to one block.
- Many clinics belong to one block.
- A road connects two blocks, and a block can be connected to many roads.
- Many parks belong to one block.
- Many heads can be assigned to one block.
- Each block has exactly one school.

Triggers:

1. **Validates CNIC Format**
    a. Staff
    b. Head
    c. SecurityStaff
    d. Shops
    e. House
    f. Mosque
    g. School
    h. Parks
    i. Clinic

2. **Roles Salary Validation**
    a. Staff
    b. Construction
    c. Schools

3. **Single Owner Trigger**
    a. Shops

4. **Prevents Owner Changes**
    a. Shops

5. **CNIC Uniqueness**
    a. Staff
    b. Head
    c. SecurityStaff
    d. Shops
    e. House
    f. Mosque
    g. School
    h. Parks
    i. Clinic

Views:

1. **StaffByPost**
2. **ConstructionWorkersByRole**
3. **BlockHeads**
4. **HeadContactInfo**

5. **SecuritySchedule**
6. **ShopsByBlock**
7. **MosqueStaff**
8. **HouseOwners**
9. **BlockResidents**
10. **All Employees**
11. **EmergencyContacts**
12. **RoleDistribution**

Functionalities:

1. Member Management

- Registration and Profile Management: Add and manage member details, including personal information, contact details, and apartment/unit numbers.
- Ownership/Rental Details: Track whether a member owns or rents a property in the society.

2. Financial Management

- Maintenance Fee Tracking: Calculate and track monthly/annual maintenance fees.
- Invoice and Payment Management: Generate invoices for residents and allow tracking or updating payment statuses.
- Penalty Calculation: Apply penalties for late payments automatically.
-

3. Facility Management

- Amenity Booking: Enable members to book facilities like a clubhouse, gym, or banquet hall.
- Schedule Management: Track facility usage schedules to avoid conflicts.

4. Complaint and Request Handling

- Complaint Logging: Residents can log maintenance or other complaints.
- Status Updates: Admin can update the status of complaints and communicate resolution times.
- Service Requests: Handle requests such as additional services or repairs.

5. Communication Tools

- Announcements and Notices: Broadcast important updates, like maintenance schedules or meetings.
- Member Notifications: Notify residents of upcoming events, dues, or facility bookings via email/SMS.

6. Security Management

- Visitor Management: Record and monitor visitor entries and exits.
- Vehicle Tracking: Maintain a record of registered vehicles.
- Gate Pass Generation: Issue gate passes for visitors or deliveries.

7. Meeting and Voting System

- Meeting Scheduling: Schedule and notify members about society meetings.
- Online Voting: Facilitate electronic voting for society's decisions.

8. Document Management

- Storage: Maintain digital copies of important documents like legal papers, maintenance contracts, or meeting minutes.
- Access Control: Restrict document access based on roles (e.g., admin-only documents).

9. Role-Based Access Control

- Different user roles like Admin, Resident, Maintenance Staff, and Security Staff can have tailored access to the system functionalities.

Conclusion

Implementing the **Housing Society Management System** using SQL will provide a **secure, scalable, and efficient** solution for managing residents, financial transactions, and security operations. With proper indexing, stored procedures, and access control, this system will ensure seamless operations for housing societies.