**UNIVERSITY INSTITUTE OF COMPUTING**

**CASE STUDY REPORT**

**ON**

**PARTICULAR CASE STUDY**

Program Name: BCA

Subject Name/Code: Database Management System (23CAT-251)

**Submitted by: Submitted to:**

**Name:** Sukhman Singh **Name: Mr. Arvinder Singh**

**UID: 23BCA10709** **Designation: Professor**

**Section:4’a’**

ABSTRACT

* Introduction:

**📘 Introduction: Hospital Patient Data Management– Database Project**

In modern healthcare, managing patient data efficiently and securely is crucial for delivering high-quality medical services. Hospitals deal with vast amounts of patient-related information daily — from personal details and medical history to appointments, prescriptions, and billing records. Traditional manual record-keeping systems are prone to data redundancy, inconsistency, and inefficiency, which can compromise patient care and decision-making.

A Database Management System (DBMS) offers a structured and reliable solution for organizing, storing, and retrieving patient information. By using a relational DBMS, hospitals can ensure accurate data entry, reduce duplication, enforce privacy and security, and enable fast access to critical medical data. This project presents the design of a Hospital Patient Data Management System using ER diagrams, schema definitions, and relationship mappings, showcasing how modern databases simplify healthcare information management.

* Technique:

**1. Relational Database Model**The hospital uses a relational database to store data in tables with rows and columns. Each table focuses on one entity (like Patient, Doctor, Appointment) which makes it easy to organize, update, and retrieve data without duplication or confusion.

**2. Entity-Relationship (ER) modelling**  
before building the database, the system is planned using an ER diagram. This diagram shows entities (like Patient, Doctor) and how they are linked (such as "Patient books Appointment with Doctor"). It helps both developers and non-technical users understand the structure of the hospital database.

**3. Normalization**Normalization is the process of breaking large, complicated tables into smaller ones to avoid data repetition and maintain data consistency. For example, a patient’s information is stored only once in the Patient table, and linked to appointments or prescriptions rather than repeating their name and details everywhere.

**4. Primary and Foreign Keys**A Primary Key is a unique ID for each table (e.g., Patient ID for the Patient table).  
A Foreign Key is used to link tables together. For example, Patient ID in the Appointment table connects each appointment to the correct patient. This helps keep the database organized and prevents errors.

**5. Structured Query Language (SQL)**SQL is the language used to interact with the database. It allows hospital staff to search for specific patient data, generate reports, update records, or delete old data — all with clear, structured commands.

**6. Constraints Enforcement**to prevent incorrect or incomplete data entry, rules (called constraints) are added to the database. For example:

* NOT NULL ensures a required field is never left blank.
* UNIQUE makes sure no two patients have the same ID.
* CHECK validates input data like age, phone number formats, etc.

**7. Role-Based Access Control**In a hospital, not every employee should have access to all data. The system assigns roles:

* Doctors can see patient records.
* Nurses can update medical observations.
* Admins handle billing and appointments.  
  This keeps sensitive data secure and prevents misuse.

**8. Backup and Recovery Mechanisms**In case of system failure, hacking, or accidental deletion, the database system automatically creates backup copies. This ensures that patient records can be restored, so no data is permanently lost.

**9. Data Security and Privacy Measures**the hospital database uses encryption and login authentication so that only authorized users can access or change sensitive information. This prevents unauthorized people from viewing or tampering with patient medical records, maintaining confidentiality.

* System Configuration:

🌐 System Configuration refers to the hardware, software, database, and networking setup that supports the functioning of the system.

Here’s what’s typically used:

🔹 **1. Database Management System (DBMS)**   
• **Software:** MySQL, PostgreSQL, Oracle, or SQL Server.  
• **Configuration Settings:**   
  o max\_connections — sets the maximum number of users that can connect to the database at once.  
  O innodb\_buffer\_pool\_size — allocates memory for data caching to speed up performance.  
  O query\_cache\_size — improves performance for read-heavy operations by storing query results.  
• **Purpose:** Stores patient data, handles relationships, enforces data integrity, and processes queries.

🔹 **2. Server Environment**  
• **Operating System:** Linux (Ubuntu/CentOS) or Windows Server.  
• **Web Server:** Apache or NGINX (if the system includes a web interface).  
• **Host Machine Configuration:**   
  o CPU: Quad-Core or higher for better performance.  
  O RAM: 8GB minimum — more is recommended for high traffic hospitals.  
  O Storage: SSD drives with at least 100GB for smooth database operations.  
• **Purpose:** Hosts the database and application backend; ensures stable access and performance.

🔹 **3. Application Layer (if using full-stack)**   
• **Languages:** Python (Flask, Django), PHP (Laravel), Java (Spring Boot), or Node.js.  
• **Framework Pattern:** Uses Model-View-Controller (MVC) architecture for organized code.  
• **API Server:** RESTful APIs or Graph used for smooth communication between frontend and backend.  
• **Purpose:** Handles business logic, user authentication, data validation, and transaction control.

🔹 **4. Frontend (User Interface)**   
• **Languages:** HTML, CSS, and JavaScript for creating web pages.  
• **Frameworks:** React.js, Angular, or Vue.js for interactive dashboards and forms.  
• **Purpose:** Displays data visually for hospital staff and patients through dashboards, forms, and reports.

🔹 **5. Network Configuration**  
• **IP Setup:** Static IP for stable server identification, protected with firewalls.  
• **Port Settings:**   
  o 3306 for MySQL database connections.  
  O 5432 for PostgreSQL database connections.  
  O 80/443 for HTTP/HTTPS web access.  
• **Secure Protocols:** HTTPS for web security, SSL/TLS for data encryption, VPN for remote access.  
• **Purpose:** Ensures safe and fast data exchange between clients and the server.

🔹 **6. Security Configuration**  
• **Access Control:** Roles assigned to users with GRANT and REVOKE permissions in SQL.  
• **Backups:** Automatic daily backups using MySQL dump or pg\_dump tools.  
• **Firewall Settings:** Allows only necessary ports and trusted IP addresses.  
• **Data Encryption:** Data is encrypted both while stored and while moving over the network.  
• **Monitoring Tools:** Use of tools like Fail2Ban, UFW, Zabbix, and Nations to monitor threats and system health.

🔹 **7. Backup and Recovery**  
• **Backup Scripts:** Scheduled backups using automated tasks like corn jobs.  
• **Recovery Plan:** Quick recovery using full database dump files or replication.  
• **Redundancy:** Uses RAID systems or cloud-based replication (like AWS RDS Multi-AZ) for extra safety.

🔹 **8. Cloud Configuration (if deployed on cloud)**   
• **Cloud Providers:** Amazon Web Services (AWS), Microsoft Azure, or Google Cloud.  
• **Common Services:**   
  o Amazon RDS for database hosting.  
  O EC2 for backend application servers.  
  O S3 for storing backup files securely.  
• **Features:** Auto-scaling for handling variable traffic, load balancing for stability, and Cloud Watch for system monitoring.

* INPUT:

In the context of the Hospital Patient Data Management System, the INPUT refers to all the data and information that is entered into the system by users, employees, or other services.

Here’s a categorized list of inputs used in the system:

**1. Patient Information**

* First Name
* Last Name
* Date of Birth
* Gender
* Phone Number
* Address
* Email (optional)

**2. Doctor Information**

* First Name
* Last Name
* Specialization
* Phone Number
* Email

**3. Appointment Details**

* Appointment Date and Time
* Patient ID
* Doctor ID
* Reason for Appointment
* Status of Appointment (Scheduled, Completed, Cancelled, etc.)

**4. Medical Records**

* Diagnosis
* Treatment Given
* Date of Entry
* Patient ID (linked to the Patient Table)

**5. Prescription Information**

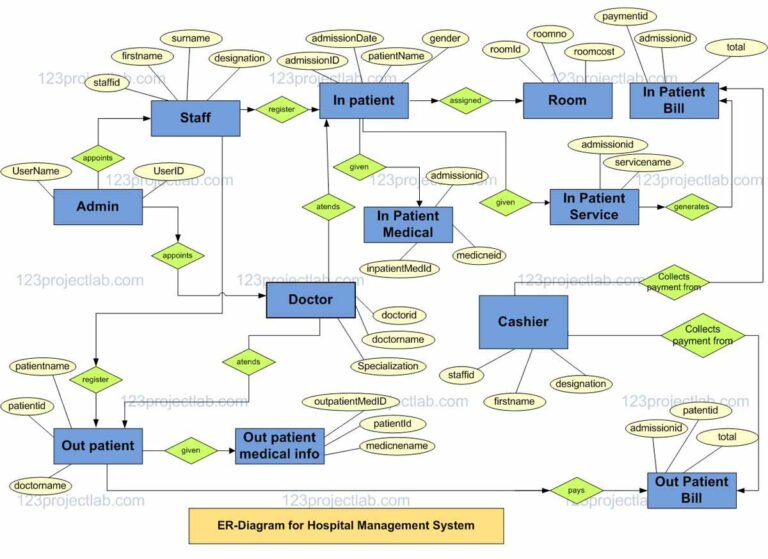
* Prescription ID
* Medicine Name
* Dosage Instructions
* Date of Prescription
* Patient ID
* Doctor ID

**6. Billing Information**

* Bill ID
* Patient ID
* Amount
* Payment Date
* Payment Method (e.g., Cash, Credit Card, Insurance)
* Billing Status (Paid, Pending, Overdue)

**7. User Authentication Inputs**

* Username
* Password
* Role (Doctor, Nurse, Admin, etc.)
* ER DIAGRAM:



* TABLE REALTION:

In a **Hospital Patient Data Management System**, **table relations** refer to the way different tables (entities) in the database are connected or associated with each other. These relationships ensure that data is consistently linked across various parts of the system, which is crucial for maintaining data integrity and accuracy.

Here are the major table relationships:

**1. One-to-Many Relationship (1: N)**

* Definition: A record in one table can be associated with multiple records in another table, but each record in the second table is linked to only one record in the first table.
* Example:
  + Patient to Appointment: One patient can have many appointments, but each appointment is associated with only one patient.
  + Doctor to Appointment: One doctor can have many appointments, but each appointment is linked to only one doctor.

**SQL Representation:**

* The Patient table has a one-to-many relationship with the Appointment table, using Patient ID as the Foreign Key in the Appointment table.
* Similarly, the Doctor table has a one-to-many relationship with the Appointment table, using Doctor ID as the Foreign Key in the Appointment table.

**2. Many-to-One Relationship (N: 1)**

* Definition: Multiple records in one table can be linked to a single record in another table.
* Example:
  + Appointment to Doctor: Multiple appointments can be associated with a single doctor.
  + Medical Record to Patient: Multiple medical records can belong to a single patient.

**SQL Representation:**

* In the Appointment table, Patient ID and Doctor ID act as foreign keys. Each appointment is related to one patient and one doctor, forming a many-to-one relationship from Appointment to Patient and Doctor.

**3. One-to-One Relationship (1:1)**

* Definition: Each record in one table is associated with exactly one record in another table.
* Example:
  + Patient to Medical Record: Each patient has one medical record, which can be updated or linked to their ongoing treatment.
  + Patient to Billing: Each patient typically has one billing record per treatment session or hospital visit.

**SQL Representation:**

* The Patient table and Medical Record table can have a one-to-one relationship where Patient ID is the Primary Key in Patient and also acts as a Foreign Key in Medical Record.

**4. Many-to-Many Relationship (M: N)**

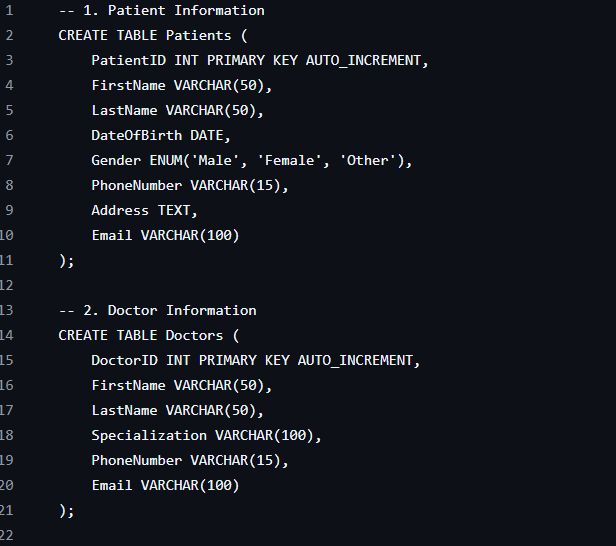
* Definition: A record in one table can be associated with multiple records in another table, and vice versa.
* Example:
  + Doctor to Prescription: A doctor can prescribe medications to multiple patients, and each patient can have prescriptions from multiple doctors.

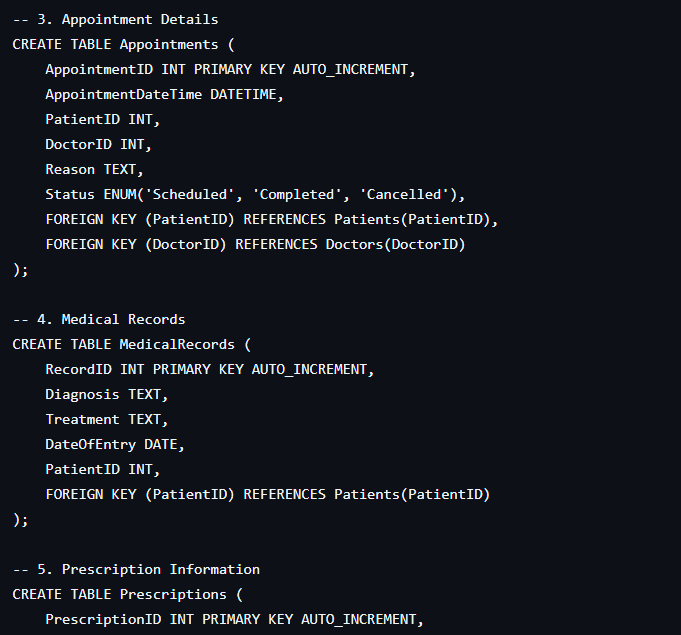
**SQL Representation:**

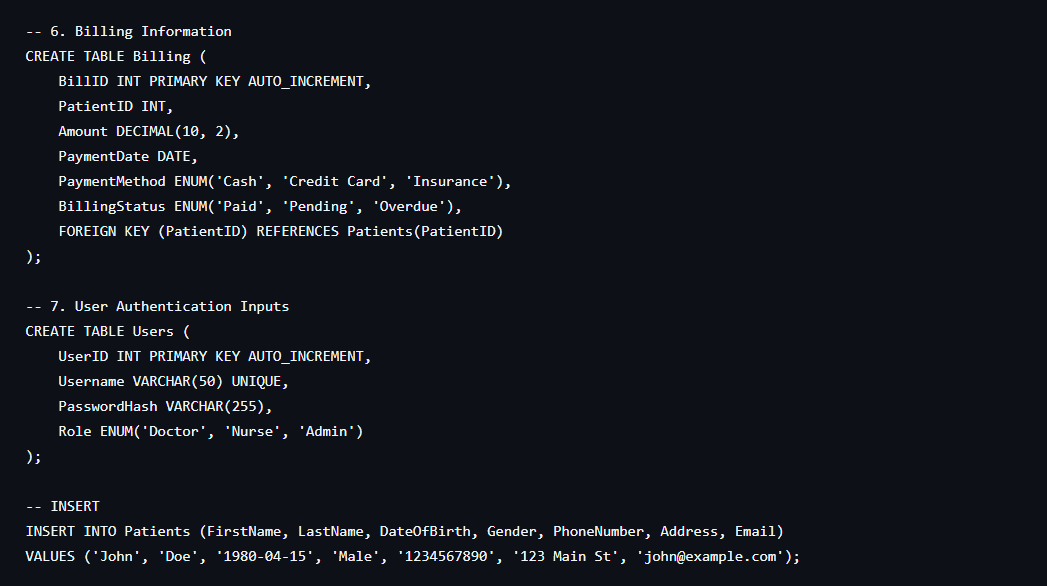
* To represent a many-to-many relationship, we create a junction table (also known as an associative table). In this case, the Prescription table acts as the junction between Doctor and Patient.
* The Prescription table will have Doctor ID and Patient ID as foreign keys to establish the relationship between doctors and patients.
* TABULAR FORMAT:

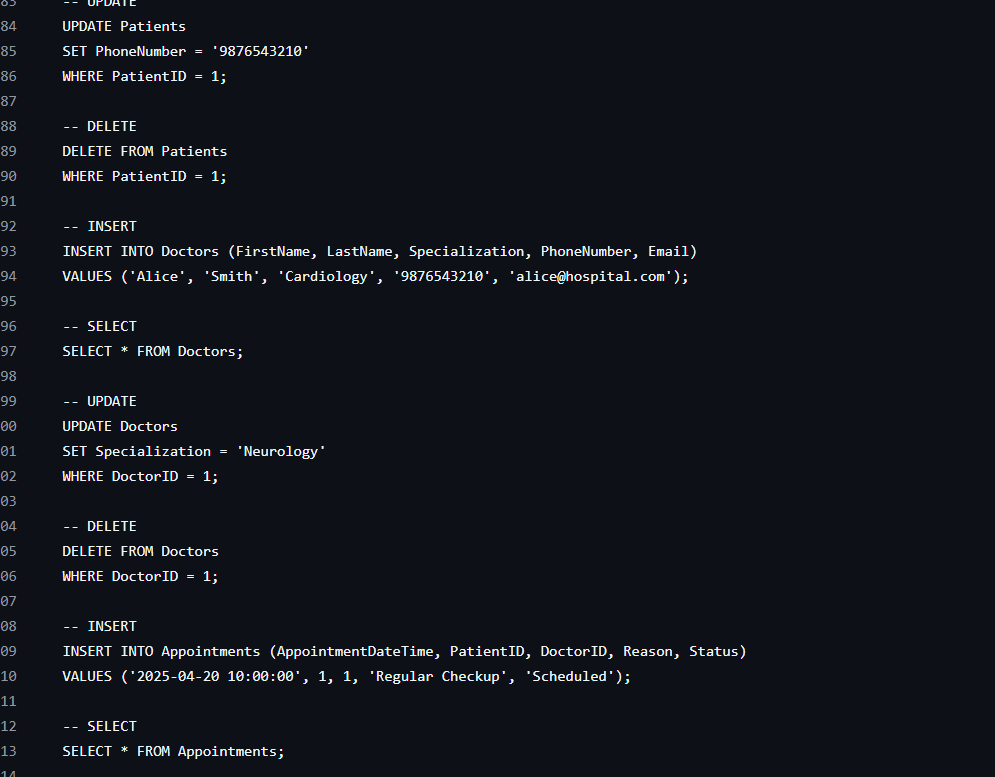
| **Table Name** | **Attribute Name** | **Data Type** | **Primary Key (PK)** | **Foreign Key (FK)** | **Constraints** |
| --- | --- | --- | --- | --- | --- |
| **Patient** | Patient ID | INT | ✅ Primary Key | None | NOT NULL, UNIQUE |
|  | First Name | VARCHAR(100) |  |  | NOT NULL |
|  | Last Name | VARCHAR(100) |  |  | NOT NULL |
|  | DOB | DATE |  |  | NOT NULL |
|  | Gender | VARCHAR(10) |  |  | NOT NULL |
|  | Phone | VARCHAR(15) |  |  | NOT NULL, UNIQUE |
|  | Address | TEXT |  |  | NOT NULL |
| **Doctor** | Doctor ID | INT | ✅ Primary Key | None | NOT NULL, UNIQUE |
|  | First Name | VARCHAR(100) |  |  | NOT NULL |
|  | Last Name | VARCHAR(100) |  |  | NOT NULL |
|  | Specialization | VARCHAR(100) |  |  | NOT NULL |
|  | Phone | VARCHAR(15) |  |  | NOT NULL, UNIQUE |
|  | Email | VARCHAR(100) |  |  | NOT NULL, UNIQUE |
| **Appointment** | Appointment ID | INT | ✅ Primary Key | Patient ID (FK) | NOT NULL |
|  | Patient ID | INT |  | Patient.PatientID (FK) | NOT NULL, FOREIGN KEY |
|  | Doctor ID | INT |  | Doctor.DoctorID (FK) | NOT NULL, FOREIGN KEY |
|  | AppointmentDate | DATETIME |  |  | NOT NULL |
|  | Reason | TEXT |  |  | NOT NULL |
|  | Status | VARCHAR(50) |  |  | NOT NULL |
| **MedicalRecord** | RecordID | INT | ✅ Primary Key | PatientID (FK) | NOT NULL |
|  | PatientID | INT |  | Patient.PatientID (FK) | NOT NULL, FOREIGN KEY |
|  | Diagnosis | TEXT |  |  | NOT NULL |
|  | Treatment | TEXT |  |  | NOT NULL |
|  | DateOfEntry | DATE |  |  | NOT NULL |
| **Prescription** | PrescriptionID | INT | ✅ Primary Key | PatientID (FK), Doctor ID (FK) | NOT NULL |
|  | PatientID | INT |  | Patient.PatientID (FK) | NOT NULL, FOREIGN KEY |
|  | Doctor ID | INT |  | Doctor.DoctorID (FK) | NOT NULL, FOREIGN KEY |
|  | Date Issued | DATE |  |  | NOT NULL |
|  | Medicine Details | TEXT |  |  | NOT NULL |
|  | Dosage Instructions | TEXT |  |  | NOT NULL |
| **Billing** | Bill ID | INT | ✅ Primary Key | PatientID (FK) | NOT NULL |
|  | PatientID | INT |  | Patient.PatientID (FK) | NOT NULL, FOREIGN KEY |
|  | Amount | DECIMAL(10, 2) |  |  | NOT NULL |
|  | Payment Date | DATE |  |  | NOT NULL |
|  | Payment Method | VARCHAR(50) |  |  | NOT NULL |
|  | Status | VARCHAR(50) |  |  | NOT NULL |

* TABLE CREATION:

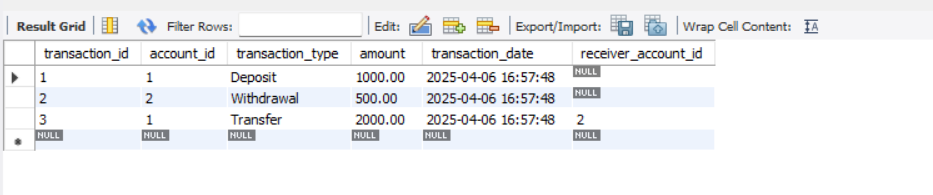


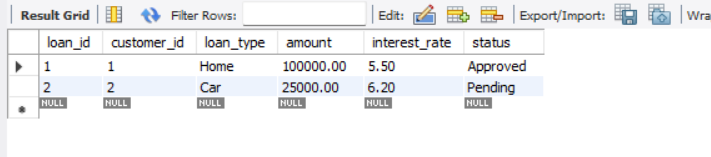






* SQL QUERIES WITH OUTPUT:





* SUMMARY:

### **Summary of Hospital Patient Data Management System**

In a **Hospital Patient Data Management System**, the goal is to efficiently manage and store data related to patients, doctors, appointments, medical records, prescriptions, and billing information. This is crucial for improving patient care, enhancing operational efficiency, and ensuring data security.

Here’s a concise summary of the system:

#### **1. Problem Statement:**

* **Data Redundancy**: Multiple records of the same patient are created across departments, leading to duplication.
* **Inconsistent Data**: Information like address or medical history may be updated in one department but not in others.
* **Data Retrieval Challenges**: Doctors and staff spend time manually searching through files, which delays access to critical patient data.
* **Limited Security**: Sensitive patient data is accessible by unauthorized personnel.
* **Scalability Issues**: As the hospital grows, managing data manually becomes increasingly complex and unmanageable.

#### **2. DBMS Solution (Database Management System):**

* **Centralized Data Storage**: All patient and medical data is stored in a secure, centralized database, reducing duplication and improving efficiency.
* **Data Integrity**: Ensures consistency and eliminates redundancy through relational database design.
* **Quick Query Access**: Doctors and staff can quickly retrieve patient histories and other data using efficient queries.
* **Role-based Access Control**: Sensitive data is only accessible to authorized personnel, enhancing privacy and security.
* **Backup and Recovery**: The system includes automatic backups, protecting the hospital from data loss.

#### **3. Schema Design (Tables and Relationships):**

* The system consists of **multiple tables** such as Patient, Doctor, Appointment, MedicalRecord, Prescription, and Billing. Each table contains relevant attributes, such as PatientID, Doctor ID, AppointmentDate, etc.
* **Primary Keys (PK)**: Ensure unique identification for each record.
* **Foreign Keys (FK)**: Establish relationships between tables (e.g., PatientID in the Appointment table links to the Patient table).
* **Constraints**: Ensure data integrity, uniqueness, and non-null ability for key fields like PatientID and Doctor ID.

#### **4. Key Relationships:**

* **Patient ↔ Appointment**: One-to-many relationship where a patient can have multiple appointments.
* **Doctor ↔ Appointment**: One-to-many relationship where a doctor can see multiple patients.
* **Patient ↔ MedicalRecord**: One-to-one relationship where each patient has a single medical record.
* **Doctor ↔ Prescription**: One-to-many relationship where a doctor prescribes medication for multiple patients.
* **Patient ↔ billing**: One-to-many relationship where a patient can have multiple billing records.

#### **5. Example Use Cases:**

* A **doctor** can check the medical history of a **patient** by querying the system for their MedicalRecord.
* A **patient** can make an **appointment** with a **doctor**, which is stored in the Appointment table.
* **Billing** records are generated after each appointment or treatment, linking the **Patient** and **Billing** tables.

#### **6. Benefits:**

* **Efficiency**: Data is quickly accessible, improving decision-making and patient care.
* **Accuracy**: Reduces human errors in patient data management.
* **Security**: Ensures that only authorized personnel have access to sensitive data.
* **Scalability**: The system can handle increasing amounts of patient data as the hospital grows.

This relational database management system addresses all the core issues in managing hospital patient data and improves the overall workflow, security, and quality of patient care.

* CONCLUSION:

### **Conclusion of Hospital Patient Data Management System**

In conclusion, the **Hospital Patient Data Management System** is a vital solution for addressing the growing challenges faced by healthcare providers in managing vast amounts of patient information. The system leverages a **Relational Database Management System (RDBMS)** to streamline data storage, improve accessibility, and enhance the security and integrity of patient records.

By centralizing patient data into structured tables and establishing clear relationships through **Foreign Keys**, the system eliminates issues such as data redundancy and inconsistency, which are common in manual systems. Furthermore, the use of **Primary Keys**, **role-based access control**, and **data encryption** ensures that patient information remains secure and accessible only to authorized personnel, maintaining high standards of privacy and confidentiality.

The **real-time query access** provided by the system ensures that doctors and hospital staff can retrieve patient histories, treatment details, and billing information quickly and efficiently, leading to better decision-making and enhanced patient care. Additionally, **automated backup and recovery mechanisms** safeguard critical patient data against loss due to system failures or other unforeseen events.

Ultimately, the **Hospital Patient Data Management System** not only increases operational efficiency within the hospital but also improves the overall quality of healthcare delivery. It offers scalability, security, and ease of use, making it an essential tool for modern healthcare institutions. As the healthcare sector continues to evolve, adopting such systems will be crucial for managing patient data effectively and ensuring the smooth running of healthcare services.

The implementation of this system is a step forward in transforming hospital operations, reducing human error, and delivering better, faster, and more secure medical services.

🎯 Final Thought:   
The implementation of a Hospital Patient Data Management System marks a significant leap forward in the healthcare sector, offering a transformative solution for the challenges associated with manual patient record-keeping. By adopting a Relational Database Management System (RDBMS), hospitals can centralize patient data, streamline workflows, enhance data accuracy, and ensure better security and privacy.

This system not only improves operational efficiency but also significantly contributes to better patient care by providing doctors and healthcare staff with quick and easy access to accurate, real-time patient information. The ability to securely manage and access patient records allows healthcare providers to make more informed decisions, minimize errors, and respond faster to patient needs.

As technology continues to shape the future of healthcare, integrating robust systems like these will be critical to maintaining the highest standards of care, improving operational workflows, and enhancing overall patient satisfaction. Ultimately, a well-structured Hospital Patient Data Management System ensures that healthcare institutions can meet both current and future demands, providing a foundation for growth, innovation, and excellence in patient care.



**https:/ /github.com/sukh12797/Hospital-Patient-Data-Management-System/tree/main**