CIS 552-01: Database Design (2024 Spring) Mid-Term Project Report

**Submitted by:**

Trina Xavier 02102403 84

**Group 13**

**Members:**  Supreeth Mohan 02036259 49

Sandeep Kasiraju 02084046 29

Trina Xavier 02102403 84

**Due date**: 03/11/2024

----------------------------------------------------------------------------------------------------------------

**Part 1: Planned Database-driven Application requirements /use-case(s)**

The planned system requirement for the Hospital Management System, as outlined in our project proposal, involves creating a comprehensive database-driven application to streamline hospital operations. The system aims to enhance patient care, optimize resource allocation, ensure data accuracy, and improve operational efficiency within the hospital setting. We have explained in detail about requirements and use cases in the proposal, so we have given a few key points here.

The key use-cases the system will implement include:

1. Patient Management: This involves storing and managing patient details, medical history, and insurance information. It facilitates efficient retrieval, updating, and tracking of patient records.

2. Appointment Scheduling: The system will manage appointment times, link patients with doctors, and maintain scheduling statuses to prevent conflicts or double bookings.

3. Medical Records Management: It will organize electronic health records, diagnoses, prescriptions, treatment plans, and lab results, ensuring easy querying and retrieval.

4. Inventory and Resource Management: This feature will handle hospital inventory, track medical supplies, equipment, and pharmaceuticals, and manage resources like hospital beds and operating rooms.

5. Billing and Insurance Management: It will process billing information and insurance claims, handle financial transactions, and provide reporting for auditing and analysis.

6. Staff Management: The system will store information on hospital staff, including roles, schedules, and evaluations, and manage hierarchical structures within the staff system.

7. Communication and Collaboration: It will facilitate secure communication among healthcare providers and integrate with external systems while upholding data integrity and security.

8. Analytics and Reporting: This functionality will support the generation of analytical reports and data visualization to aid decision-making and performance monitoring.

Furthermore, the system will include personalized patient portals with strong security measures and compliance with regulations such as HIPAA.

In conclusion, the Hospital Management System is designed to be a relational database management system that supports the critical operations of a healthcare facility, from patient care to backend administrative functions, all through programmatic access to a well-structured database.

**Part 2: Entity-Relationship Diagram**

**A diagram of a database

Description automatically generated**

This ER diagram meticulously outlines the interconnections between various entities that make up the hospital's workforce, the patients they serve, the administrative structure, and the resources managed, ensuring a comprehensive framework for effective data management and operational workflow in the hospital management system.

**Part 3: Corresponding Relational Schema**

A screenshot of a computer

Description automatically generated

This streamlines various aspects of hospital operations. At the core of this schema is the `Employee` table which has relationships with several other tables such as `Doctor`, `Nurse`, and `General`. These tables inherit the primary key `EmployeeID` as a foreign key from the `Employee` table, signifying that doctors, nurses, and other general employees are all categorized under the employee umbrella. Each employee is linked to a specific department as indicated by the `Department Number (FK)` foreign key in their respective tables. Moreover, the schema includes an `Appointment` table that stores appointments made by patients with doctors, and it is linked to both the `Patient` and `Doctor` tables through their respective IDs.

Also manages patient information and interactions with the medical facility through tables like `Patient` and `Medical Records`. The `Patient` table encompasses comprehensive personal and contact details, insurance information, and medical history summaries, ensuring that the patient's healthcare journey is meticulously documented. Medical records are directly associated with patients and doctors, providing a detailed history of diagnoses and visits. Furthermore, the schema delineates the management hierarchy with a `Hospital Admin` table, illustrating the administrative roles and their associated details. A `Resource Management` table exists to oversee hospital resources, ensuring their optimal allocation and availability. This relational schema effectively organizes the interrelated aspects of hospital management into a cohesive and efficient structure.

**Part 4: Corresponding DB Implementation**

To begin with, we have created the Database named **‘Hospital Management System’.**

A screenshot of a computer

Description automatically generated

Executed the Create Queries for each entity, below are few screenshots for the same.

A screenshot of a computer

Description automatically generated

**Part 5: Initial Populating of Database**

Next, we inserted records for each table, below attached are the few screenshots for the same.

A screenshot of a computer

Description automatically generated

The below two screenshots show select queries successfully running by retrieving the data from the tables of initial populated data into hospital management database.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Part 6: Initial Demonstration of DB Querying (CRUD Operations)**

* **Create/Insert Operation:**

Initially, there are 6 rows, which were executed for insertion of new records to perform the insert operation.

A screenshot of a computer

Description automatically generated

Now, we executed the **Insert Query** to insert two more records and performed the insert operation and the query executed successfully.

A screenshot of a computer

Description automatically generated

In the below Image you can see the two new records, with **PatientID 17 and 18** have been added.

A screenshot of a computer

Description automatically generated

* **Read and Update Operation:**

We have selected the appointment table, where there are 6 records which we get to know through **READ operation**, which were initially inserted for populating the DB.

A screenshot of a computer

Description automatically generated

Now, we will update the Appointment for PatientID is 7, and perform the **UPDATE operation.**

A screenshot of a computer

Description automatically generated

After Updating, we can witness that, the PatientID - 7 has the DateTime column changed as given in the **UPDATE query**.

A screenshot of a computer

Description automatically generated

**Delete Operation**:

To perform delete operation, we have used **between** operator to delete Patient appointments from PatientID 6 to 8. So, a total of 3 records will be deleted and have been deleted as seen in the screen capture.

A screenshot of a computer

Description automatically generated

As you can see in the below image, we just have 3 records after deleting the 3 records out of 6 records.

A screenshot of a computer

Description automatically generated

**7. Stored procedure and Embedded Query:**

* **Stored Procedure 1:**

Creation of Stored procedure for inserting a new patient into the patient table.

A screenshot of a computer

Description automatically generated

* **Creation of Trigger:**

After creating the stored procedure for patient, a trigger has been created under patient table, to check whether the DoctorID to be inserted in the patient table is existing in Doctor table or not, If yes, then the record will be inserted with no errors, else, the trigger will be triggered and throws an output sat=ying ‘DoctorID is not present, please enter the correct DoctorID’.

A screenshot of a computer

Description automatically generated

Below are the details of all Patients in the Patient table which were inserted previously.

A screenshot of a computer

Description automatically generated

**Execution of ‘AddNewPatient’ Trigger and Stored procedure 1:**

This is an incorrect query to demonstrate the trigger operation. We insert the DoctorID, which is a foreign key in Patient Table. When the DoctorID provided is not present in the Patient table, the Trigger created in patient table gets triggered and prints the ‘Doctor ID does not exist’ statement. So, the user will get to know the Doctor ID is not correct and should enter the correct DoctorID.

A screenshot of a computer

Description automatically generated

Correct query is executed with the DoctorID that is present in the Doctor table. We can see from the below screen capture, that the **StoredProcedure** has been executed successfully and the record has been inserted.

A screenshot of a computer

Description automatically generated

The output for the above insertion is displayed in the below screen capture, which has PatientID 20, and DoctorID 9, which has been newly inserted.

A screenshot of a computer

Description automatically generated

* **Stored Procedure 2:**

Creation of Stored procedure for inserting a new employee into the employee table.

**A screenshot of a computer

Description automatically generated**

**Execution of ‘AddNewEmployee’ Stored procedure 2:**

The second stored procedure is to add a new employee, as you can see in the below screen capture, there are 6 Employees for now.

A screenshot of a computer

Description automatically generated

Now, we try to execute an incorrect query which has Incorrect DepartmentNumber which is a foreign key in Employee table, which means a department should be present before an employee has been inserted into the table.

A screenshot of a computer

Description automatically generated

Now, when the correct query is executed, the **insertion is successful** through the Stored procedure.

A screenshot of a computer

Description automatically generated

The output of insertion of the new employee can be seen in the below screen capture, having **EmployeeID - 9.**

A screenshot of a computer

Description automatically generated

**Conclusion Remarks:**

Ultimately, through the execution and application of this project, we gained insights into the process of constructing a Database ER Diagram and Schema, and then applying them within MSSQL. Additionally, we familiarized ourselves with the practical application of CRUD operations, Stored Procedures, Functions, and Triggers in SQL, all of which play pivotal roles in real-world scenarios. This experience broadened our understanding of database management and its practical implications.

Moreover, engaging in this project enabled us to dive deeper into the nuances of crafting and overseeing databases, offering us practical insights into maneuvering through the intricate web of relational databases. Through our exploration of MSSQL capabilities, we uncovered the intricate dance between these components, witnessing firsthand how they synergize to sustain the smooth functioning of data systems. Armed with this knowledge, we are better prepared to confront the complexities of database administration and development in real-world scenarios, honing our ability to manage varied data architectures and fine-tune database efficiency across a spectrum of applications.

**References:**

* This relational schema draws upon the database design principles as outlined in lecture slides by Prof. Ashok Kumar Patel.
* Fundamentals of Database Systems (7th Edition), Authors: R. Elmasri & S. Navathe.