|  |
| --- |
| Student’s Full Name: SHREYA VIKAS KOLTE |
| Course Title: INFO 579: SQL/NoSQL Databases for Data and Information Sciences |
| Term name and year: Spring 2024 |
| Submission Week: Week 16 Assignment |
| Instructor’s Name: Prof. Nayem Rahman |
| Date of Submission: 08-05-2024 |

**Problem Statement:**

The Hospital Management System (HMS) project aims to resolve the challenges healthcare facilities face in managing critical patient information, staff allocations, and billing processes. Current systems often lack efficient data organization, resulting in delayed decision-making, billing errors, and a lack of comprehensive insights into healthcare operations. By creating a scalable database model with well-structured entities and relationships, the HMS will streamline data collection, improve patient record accessibility, and facilitate seamless management of healthcare processes.

**Data Explanation:**

* First data sheetcontains unstructured data representing individual patient details like name, contact information, birthdate, gender, address, and email. It also includes links to associated doctors and other healthcare professionals.
* The second data sheet presents normalized patient data organized into multiple columns for:
  + **Patient Information**: ID, name, contact, birth date, gender, and email.
  + **Medical History**: Highlights health conditions or surgical needs.
  + **Billing Details**: Associates each patient with a billing ID.

**Many-to-Many Relationships:**

* **Patient-Doctor (DoctorPatient Table):**

Description: A patient may consult multiple doctors, and a doctor may treat multiple patients.

* Implementation: This relationship is managed through the DoctorPatient junction table, which links PatientID to DoctorID.
* **Patient-Nurse (PatientNurse Table):**
* Description: Multiple nurses can attend to one patient, and a nurse can provide care to multiple patients.
* Implementation: The PatientNurse junction table links PatientID to NurseID.
* **Patient-Surgery (PatientSurgery Table):**
* Description: A patient may undergo multiple surgeries, and a surgery may involve multiple patients.
* Implementation: The PatientSurgery table links PatientID to SurgeryID.

**One-to-Many Relationships:**

* **Doctor-Department:**

Description: A department typically has many doctors, but a doctor is usually associated with only one department.

Implementation: The Doctor table contains a DeptID field, which is a foreign key referencing the Department table's primary key.

* **Surgery-Doctor:**

Description: Multiple surgeries can be handled by a single doctor.

Implementation: The Doctor table contains a SurgeryID field linking to the primary key in the Surgery table.

**One-to-One Relationships:**

* **Patient-Bill:**
  + Description: Each patient has a unique billing record.
  + Implementation: The Patient table has a foreign key BillID, referencing the Bill table's primary key.

The conceptual model provides a high-level overview of the entities and their relationships within the Hospital Management System (HMS).

The logical diagram expands upon the conceptual model by introducing more granular relationships and attributes. It includes primary (PK) and foreign keys (FK) that enforce relationships among entities.

The physical model is constructed using the logical design as a reference. It provides a detailed structure for each table and column within the Hospital Management System (HMS). It defines the database schema with constraints, data types, and relationships, ensuring proper data integrity and consistency.

**Table Creation and Data Insertion**

1. **Create Tables Using DDL Statements**
   * DDL (Data Definition Language): SQL statements like CREATE TABLE define the schema.
   * Tables like Patient, Doctor, Surgery, Department, and junction tables (PatientDoctor, etc.) are created based on the physical model.
2. **Implement Column Constraints and Keys**
   * Columns: Define names, data types (e.g., varchar, int), and constraints like NOT NULL.
   * Primary Key (PK): Uniquely identifies each row, like PatientID in the Patient table.
   * Foreign Key (FK): Links to another table's PK to maintain referential integrity.
3. **Insert Data into Tables**
   * DML (Data Manipulation Language): SQL INSERT statements populate tables with actual data.

**Create SQL Queries**

1. **SELECT \* from Each Table**
   * Retrieves all columns and rows from a table and orders them by primary key. This involves querying each table individually to ensure that all data is present and correctly ordered by primary key.
2. **INNER JOIN with Junction Tables**
   * Joins the main tables via a junction table to showcase a many-to-many relationship. Inner joins between tables and junction tables (e.g., Patient-Doctor relationships) reveal how many-to-many relationships are mapped, returning only matched pairs.
3. **LEFT OUTER JOIN**
   * This approach compares the results to inner joins, showing which rows would be excluded in an inner join. It compares the output with INNER JOIN behavior. A left outer join combines data from multiple tables while retaining unmatched rows from the "left" table.
4. **Single-Row Subquery**
   * Nested query that returns a single row as input for the outer query. The output is sorted by a key field. A subquery retrieves a single value, which is used to filter or match rows in an outer query. For instance, identifying a patient based on a specific billing condition.
5. **Multiple-Row Subquery**
   * Nested query that returns multiple rows to the outer query for filtering. This subquery returns a list of values used to filter rows in the outer query. It helps find specific patients that satisfy particular criteria (e.g., billing above a certain amount).
6. **Aggregate Results Using Multiple Columns**
   * Used GROUP BY to summarize data over multiple fields. This provides insights into trends and statistics. Aggregating results with functions like **COUNT**, **SUM**, etc., over multiple columns provides summaries, like the number of patients per doctor.
7. **NOT IN Operator**
   * Filters rows that are not present in a subquery result set. Excludes rows based on the results of a subquery, which helps identify patients or doctors who are not associated with specific conditions.
8. **CASE Statement**
   * Implements conditional logic to categorize data based on specified criteria. The **CASE** statement categorizes rows based on conditional logic. This could include grouping patients into billing tiers.
9. **NOT EXISTS Operator**
   * Tests for the non-existence of rows that meet the subquery's condition. Checks for the non-existence of a relationship in the subquery, helping identify doctors without any patients.
10. **NOT NULL Operator in Subquery**
    * Ensures that rows returned from a subquery have non-null values in specific columns. Filters rows to only include those with non-null values in specific columns, ensuring that only patients with complete records are retrieved.

**Real-World Implications of the Project**

The Hospital Management System (HMS) project is designed to provide a streamlined and efficient way to handle the critical operations of a healthcare facility.

1. **Patient:**
   * **Real-World Implication:** Central to any healthcare facility, patients are at the heart of all activities. Maintaining comprehensive patient data is crucial for accurate diagnoses, effective treatments, and personalized care plans.
   * **Relationships:** Links with all other entities, including doctors (via many-to-many relationships), surgeries, departments, and billing. This ensures seamless coordination of care and efficient patient management.
2. **Doctor:**
   * **Real-World Implication:** Doctors play a pivotal role in delivering quality healthcare. By maintaining up-to-date records of doctors' specialties, departments, and patient assignments, the system ensures that patients are treated by the appropriate professionals.
   * **Relationships:** Connected to patients, surgeries, and departments, emphasizing a multidisciplinary approach to treatment.
3. **Nurse:**
   * **Real-World Implication:** Nurses are indispensable in patient care, providing continuous support before, during, and after treatments. Their records ensure accurate tracking of which nurses have attended to which patients.
   * **Relationships:** Tied to patients through many-to-many junctions, highlighting their involvement in multiple patients' care.
4. **Surgery:**
   * **Real-World Implication:** Surgeries are often complex and involve collaboration between various healthcare professionals. Detailed records of surgical procedures help coordinate teams, reduce errors, and improve patient outcomes.
   * **Relationships:** Associated with patients via many-to-many relationships and doctors through one-to-many links.
5. **Department:**
   * **Real-World Implication:** Healthcare departments specialize in specific medical fields and provide the infrastructure for specialized care. Keeping an accurate department allocation ensures patients are routed to the correct departments and receive the appropriate care.
   * **Relationships:** Connected to doctors in one-to-many relationships, as doctors typically belong to one department.
6. **Bill:**
   * **Real-World Implication:** Accurate billing is crucial for a hospital's financial health. It ensures transparency, minimizes errors, and facilitates payment processing. Each patient has a unique bill linked directly to their treatments.
   * **Relationships:** Linked one-to-one with each patient to ensure that every patient has an accurate and unique billing record.

**Efficient Coordination:** The many-to-many and one-to-many relationships ensure that patients are attended by multiple professionals efficiently while allowing collaboration across departments.

**Accurate Billing and Documentation:** A unified view of patients' treatments ensures accurate billing and documentation, reducing potential errors.

**Improved Decision-Making:** Aggregate data from relationships between doctors, nurses, departments, and patients provides management insights to allocate resources better and improve care quality.

**Enhanced Patient Experience:** The structured data model helps provide a holistic view of patient care, making follow-ups and treatment consistency easier to manage.

Overall, the Hospital Management System aims to make patient care safer, more efficient, and more transparent, ultimately improving patient outcomes and operational workflows for healthcare facilities.