2023

Database Management with SQL

O.P. Jindal Global University
SHUAIB SULEMAN
DATABASE DESIGNING FOR A HOSPITAL

Instructions

Design and develop a database for a hospital for which the details are as follows:

a) It will maintain information about all departments, such as department ID, department name, contact no., and head of the department.

b) It will maintain information about the doctors, such as Doctor ID, name, address, qualification, specialization, designation, contact no., and department ID.

c) It will maintain information about all the patients, such as patient ID, name, sex, age, date of admission, date of discharge, disease, test results, address, and contact no.

d) It will maintain information about all the nurses, such as nurse ID, name, address, qualification, designation, contact no, and department.

e) It will maintain information about all the tests that are being done in the hospital, such as test ID, test name, test result, concerned patients, and concerned doctor.

f) It will maintain information about all the wards in each department, such as ward no, department, and ward capacity.

In the design phase, we will start our design process by making assumptions and identifying entities and relationships among them. We will finish the design phase by drawing an E-R diagram which will model all the entities and relationships among them for the hospital database.

Assumptions

According to the problem statement, we are making the following assumptions:

a) A department may have many doctors, but a doctor will work in only one department.

b) A department may have many nurses, but a nurse will work in only one department.

c) A department may have multiple wards, but one ward will be attached to one department only.

d) A department may have multiple contact numbers.

e) Doctors and nurses may visit the patients admitted in the wards of different departments.

Question 1

(a) Identify the entities involved and their respective attributes. Give in tabular format. (Hint: six entities)

Solution:

Department	Department_ID (Primary Key), Department_Name, Contact_No, Head_of_Department
Doctor	Doctor_ID (Primary Key), Name, Address, Qualification, Specialization, Designation, Contact_No, Department_ID(Foreign Key)
Patient	Patient_ID (Primary key), Name, Sex, Age, Date_of_Admission, Date_of_Discharge, Disease, Test_Results, Address, Contact_No
Nurse	Nurse_ID(Primary Key), Name, Address, Qualification, Designation, Contact_No, Department_ID (Foreign Key)
Test	Test_ID (Primary Key), Test_Name, Test_Result, Patient_ID (Foreign Key), Doctor_ID (Foreign Key)
Ward	Ward_No (Primary Key), Department_ID (Foreign Key), Ward_Capacity

The primary keys are unique identifiers for each record in the entity. The foreign keys establish a link between the data in two tables. For example, the Department_ID in the Doctor entity is a foreign key to the Department_ID in the Department entity, establishing a link between a doctor and their department.

(b) Write down separately the relationships that you will identify. (Hint: eight relationships)

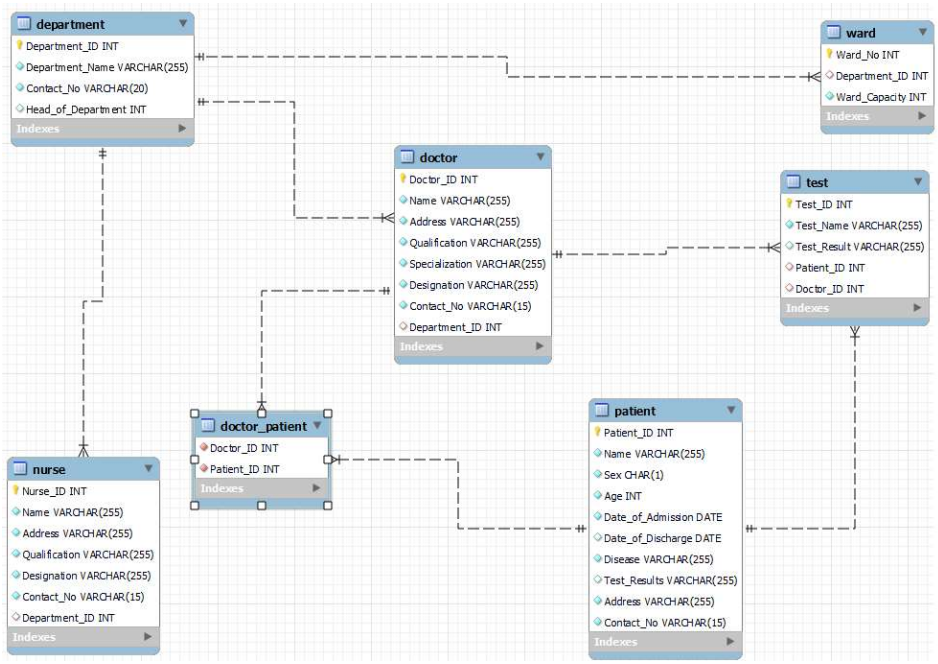
The relationships that I will identify for the hospital database:

1. Department-Doctor (one-to-many):One Department can have many Doctors, but a doctor can work in only one Department.
2. Department-Nurse (one-to-many): A department has many nurses, but a nurse belongs to only one department.
3. Department-Ward (one-to-many): A department has many wards, but a ward belongs to only one department.
4. Doctor-Patient (many-to-many): A doctor may visit many patients, and a patient may be visited by many doctors.
5. Nurse-Patient (many-to-many): A nurse may visit many patients, and a patient may be visited by many nurses.
6. Doctor-Test Relationship (many-to-many): A doctor can order multiple tests, and a test can be ordered by multiple doctors. This is a many-to-many relationship between Doctor and Test.
7. Patient-Test Relationship (one-to-many): A patient can have multiple tests, but a test is conducted for only one patient. This is a one-to-many relationship from Patient to Test.
8. Ward-Patient Relationship (one-to-many): A ward can accommodate multiple patients, and one patient can stay in one ward at a time. This is a one-to-many relationship from Ward to Patient.

Question 2

Based on entities and their relationships, draw an E-R diagram to model all the entities and relationships among them.

ER Diagram



Legend:

- Entities: These are represented by rectangles. Each entity represents a type of object in the real world that is significant to the database. In this diagram, the entities are "Department", "Doctor", "Patient", "Nurse", "Test", “Doctor_Patient” and "Ward".
- Attributes: These are represented by diamonds in their respective entities. Attributes are the properties or characteristics of entities. For example, the "Doctor" entity has attributes like "Doctor_ID", "Name", "Address", "Qualification", "Specialization", "Designation", "Contact_No", and "Department_ID".
- Primary Key: This is represented by the attribute prefixed with a 'yellow lightbulb'. A primary key is a unique identifier for a record in a table. For example, "Doctor_ID" is the primary key for the "Doctor" entity.
- Foreign Key: This is represented by the attribute prefixed with a 'purple diamond outline'. A foreign key is a field in a table that matches the primary key of another table. It is used to establish a link between the data in two tables. For example, "Department_ID" in the "Doctor" entity is a foreign key that links to the "Department_ID" primary key in the "Department" entity.
- Relationships: These are represented by lines connecting entities. The relationships are iindicate the relationship between the entities. For example, the line connecting "Doctor" and "Patient" is indicating that a doctor provides medical care to a patient.
- Cardinality: This is represented by the symbols at the ends of the relationship lines. The '||' symbol indicates "one and only one", the 'o{' symbol indicates "one or many", and the '}' symbol indicates "many". For example, the line between "Doctor" and "Department" has '||' on the "Department" end and 'o{' on the "Doctor" end, indicating that one department can employ many doctors, but each doctor works in one and only one department.

The E-R diagram visually represents the relationships between entities in the hospital database. It can be used as a guide for creating database tables and ensuring referential integrity through foreign key constraints.

Question 3

Write the SQL statements to create all tables specifying the data types for each field (Hint: 6 tables). Create all the primary keys, foreign keys, NOT NULL, UNIQUE constraints, and domains wherever applicable inside the CREATE TABLE statements.

SQL statements to create the tables in the hospital database, along with primary keys, foreign keys, NOT NULL, UNIQUE constraints, and domains:

Department Table:

```
-- Creating Department table
CREATE TABLE IF NOT EXISTS Department (
    Department_ID INT AUTO_INCREMENT PRIMARY KEY,  -- Primary Key
    Department_Name VARCHAR(100) NOT NULL,  -- Name of the Department
    Contact_No VARCHAR(20) NOT NULL UNIQUE,  -- Contact Number of the Department
    Head_of_Department VARCHAR(100) NOT NULL  -- Head of the Department
);
```

The Department table contains information on each hospital department. The 'Department ID' primary key is a unique identifier for each department. The 'Department Name' and 'Contact No' columns are not nullable, guaranteeing that each department has a unique name and contact number. The 'Head of Department' field is null if no department head is assigned and relates to the department head's ID.

Doctor Table:

```
-- Creating Doctor table
CREATE TABLE IF NOT EXISTS Doctor (
    -- Creating Doctor table
CREATE TABLE IF NOT EXISTS Doctor (
    Doctor_ID VARCHAR(20) PRIMARY KEY,  -- Primary Key
    Name VARCHAR(100) NOT NULL,  -- Name of the Doctor
    Address VARCHAR(200) NOT NULL,  -- Address of the Doctor
    Qualification VARCHAR(100) NOT NULL,  -- Qualification of the Doctor
    Specialization VARCHAR(100) NOT NULL,  -- Specialization of the Doctor
    Designation VARCHAR(100) NOT NULL,  -- Designation of the Doctor
```

```
    Contact_No VARCHAR(20) NOT NULL UNIQUE,  -- Contact Number of the Doctor
    Department_ID INT NOT NULL,  -- Foreign Key, Reference to Department_ID in Department table
    FOREIGN KEY (Department_ID) REFERENCES Department(Department_ID)
);
```

The Doctor table contains information on each hospital doctor. Doctor ID, the table's main key, is a unique identifier for each doctor. Name, Address, Qualification, Specialty, and Designation are essential fields that must be filled in. The 'Contact No' parameter is not null and unique, guaranteeing that each doctor has a unique contact number. The 'Department ID' field is a foreign key that references the 'Department' database and indicates the doctor's department of employment.

Nurse Table:

```
-- Creating Nurse table
CREATE TABLE IF NOT EXISTS Nurse (
    Nurse_ID INT AUTO_INCREMENT PRIMARY KEY,  -- Primary Key
    Name VARCHAR(20) NOT NULL,  -- Name of the Nurse
    Address VARCHAR(200) NOT NULL,  -- Address of the Nurse
    Qualification VARCHAR(255) NOT NULL,  -- Qualification of the Nurse
    Designation VARCHAR(255) NOT NULL,  -- Designation of the Nurse
    Contact_No VARCHAR(20) NOT NULL UNIQUE,  -- Contact Number of the Nurse
    Department_ID INT NOT NULL,  -- Foreign Key, Reference to Department_ID in Department table
    FOREIGN KEY (Department_ID) REFERENCES Department(Department_ID)
);
```

The Nurse table contains information on each hospital nurse. The table's primary key is Nurse ID, a unique identification for each nurse. The mandatory fields 'Name,' 'Address,' 'Qualification,' and ‘Designation ‘Must be filled in. The 'Contact No' column is not null and unique, guaranteeing that each nurse has a unique contact number. The 'Department ID' field is a foreign key that references the 'Department' database, showing the nurse's department of employment.

Ward Table:

```
CREATE TABLE IF NOT EXISTS Ward (
    WardNo INT PRIMARY KEY,
    DepartmentID INT,
```

```
    WardCapacity INT,
    FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)
);
```

The Ward table stores information about each ward in the hospital. It has a primary key, 'Ward_No', a unique identifier for each ward. The 'Department_ID' field is a foreign key referencing the 'Department' table, indicating which department the ward belongs to. The 'Ward_Capacity' field specifies the ward's capacity and cannot be null.

Patient Table:

```
-- Creating Patient table
CREATE TABLE IF NOT EXISTS Patient (
    Patient_ID VARCHAR(20) PRIMARY KEY,  -- Primary Key
    Name VARCHAR(100) NOT NULL,  -- Name of the Patient
    Sex CHAR(1) NOT NULL,  -- Sex of the Patient M or F
    Age INT NOT NULL,  -- Age of the Patient
    Date_of_Admission DATE NOT NULL,  -- Admission Date of the Patient
    Date_of_Discharge DATE,  -- Discharge Date of the Patient
    Disease VARCHAR(255) NOT NULL,  -- Disease of the Patient
    Test_Results VARCHAR(255),  -- Test Results of the Patient
    Address VARCHAR(200) NOT NULL,  -- Address of the Patient
    Contact_No VARCHAR(20) NOT NULL UNIQUE  -- Contact Number of the Patient
);
```

The Patient table contains information on each hospital patient. Patient ID, the main key, is a unique identifier for each patient. The required fields 'Name,' 'Sex,' 'Age,' 'Date of Admission,' 'Disease,' and 'Address' cannot be left blank. 'Date of Discharge' can be left blank if the patient still needs to be discharged. The 'Contact No' field is mandatory and cannot be left blank, ensuring each patient has a unique contact number. The 'Ward No' column is a foreign key that references the 'Ward' database and indicates the ward to which the patient has been admitted.

Test Table:

```
-- Creating Test table
CREATE TABLE IF NOT EXISTS Test (
```

```
    Test_ID INT PRIMARY KEY,  -- Primary Key
    Test_Name VARCHAR(200) NOT NULL,  -- Name of the Test
    Test_Result VARCHAR(255),  -- Result of the Test
    Patient_ID VARCHAR(20) NOT NULL,  -- Foreign Key, Reference to Patient_ID in Patient table
    Doctor_ID VARCHAR(20) NOT NULL,  -- Foreign Key, Reference to Doctor_ID in Doctor table
    FOREIGN KEY (Patient_ID) REFERENCES Patient(Patient_ID),
    FOREIGN KEY (Doctor_ID) REFERENCES Doctor(Doctor_ID)
);
```

The Test table contains information on every test performed at the facility. Test ID, its main key, is a unique identifier for each test. The 'Test Name' and 'Test Result' entries are mandatory and cannot be left blank. The 'Patient ID' field is a foreign key that references the 'Patient' table and identifies which patient is tested. The 'Doctor ID' field is a foreign key that references the 'Doctor' database, indicating which doctor performs the examination.

Doctor-patient table

```
-- Creating Doctor_Patient table
CREATE TABLE IF NOT EXISTS Doctor_Patient (
    Doctor_ID VARCHAR(255) NOT NULL,  -- Foreign Key, Reference to Doctor_ID in Doctor table
    Patient_ID VARCHAR(255) NOT NULL,  -- Foreign Key, Reference to Patient_ID in Patient table
    PRIMARY KEY (Doctor_ID, Patient_ID),  -- Composite Primary Key
    FOREIGN KEY (Doctor_ID) REFERENCES Doctor(Doctor_ID),
    FOREIGN KEY (Patient_ID) REFERENCES Patient(Patient_ID)
);
```

The hospital system must track doctor-patient connections, especially if a patient can be treated by numerous doctors and a doctor can treat multiple patients. We can effectively and properly track the many-to-many relationships between doctors and patients by building this Doctor_Patient table.

Question 4

Based on the tables created, write the SELECT statements to :

(a) List the names of the doctors of the "Orthopedic" department with their qualifications and contact numbers.

```
-- Query to retrieve Doctors from Orthopedic department

SELECT Doctor.Name, Doctor.Qualification, Doctor.Contact_No

FROM Doctor

JOIN Department ON Doctor.Department_ID = Department.Department_ID

WHERE Department.Department_Name = 'Orthopedic';
```

By combining the Doctor table with the Department table, this query retrieves the doctor's name, credentials, and phone number. The result is filtered based on the Orthopedic department name.

(b) List all the departments having less than three nurses.

```
-- Query to retrieve Departments with less than 3 nurses

SELECT Department.Department_ID, Department.Department_Name

FROM Department

JOIN Nurse ON Department.Department_ID = Nurse.Department_ID

GROUP BY Department.Department_ID, Department.Department_Name

HAVING COUNT(Nurse.Nurse_ID) < 3;
```

This query will return a list of departments with less than three nurses and their IDs. The statement joins the Department and Nurse tables on the Department_ID column, then groups the results by Department_ID and Department_Name and filters the groups to include only those with less than three Nurse_IDs.

(c) List all the patients with their ages, sexes, admission dates, and discharge dates who have been treated by the doctor whose ID is "D-10".

```
-- Query to retrieve patient information who are treated by doctor with ID '10'

SELECT Patient.Patient_ID, Patient.Name, Patient.Age, Patient.Sex, Patient.Date_of_Admission,
Patient.Date_of_Discharge

FROM Patient

JOIN Doctor_Patient ON Patient.Patient_ID = Doctor_Patient.Patient_ID
```

```
WHERE Doctor_Patient.Doctor_ID = 'D-10';
```

This SQL statement retrieves comprehensive information from the Patient table for patients who have been treated by the doctor with ID 'D-10'. The JOIN clause is critical in this case because it allows the query to grasp the relationship between the Patient and Doctor_Patient tables and, as a result, filter the patients based on the doctor who treated them.

(d) Find the wards with the maximum capacity.

```
--- Query to retrieve Ward with highest ward number

SELECT WardNo, WardCapacity

FROM Ward

WHERE WardCapacity = (SELECT MAX(WardCapacity) FROM Ward);
```

This query finds the maximum value in the Ward_Capacity column and selects the Ward_No and Ward_Capacity for all records where Ward_Capacity matches this maximum value.

(e) List the details of all tests conducted for the patient with ID "P-04" referred by the doctor with id "D-01".

```
-- Query to retrieve test information for patient with ID 'P-04' performed by doctor with ID 'D-01'

SELECT Test.Test_ID, Test.Test_Name, Test.Test_Result

FROM Test

JOIN Patient ON Test.Patient_ID = Patient.Patient_ID

JOIN Doctor ON Test.Doctor_ID = Doctor.Doctor_ID

WHERE Patient.Patient_ID = 'P-04' AND Doctor.Doctor_ID = 'D-01';
```

We perform a join operation between the Test table, the Patient table, and the Doctor table based on their respective foreign keys. It then filters the results based on the conditions that the Patient_ID is equal to 'P-04', and the Doctor_ID is equal to 'D-01'.