M·Sai·Sindhuja - 2023001092

DBMS Case Study

Hospital Management System

Introduction:

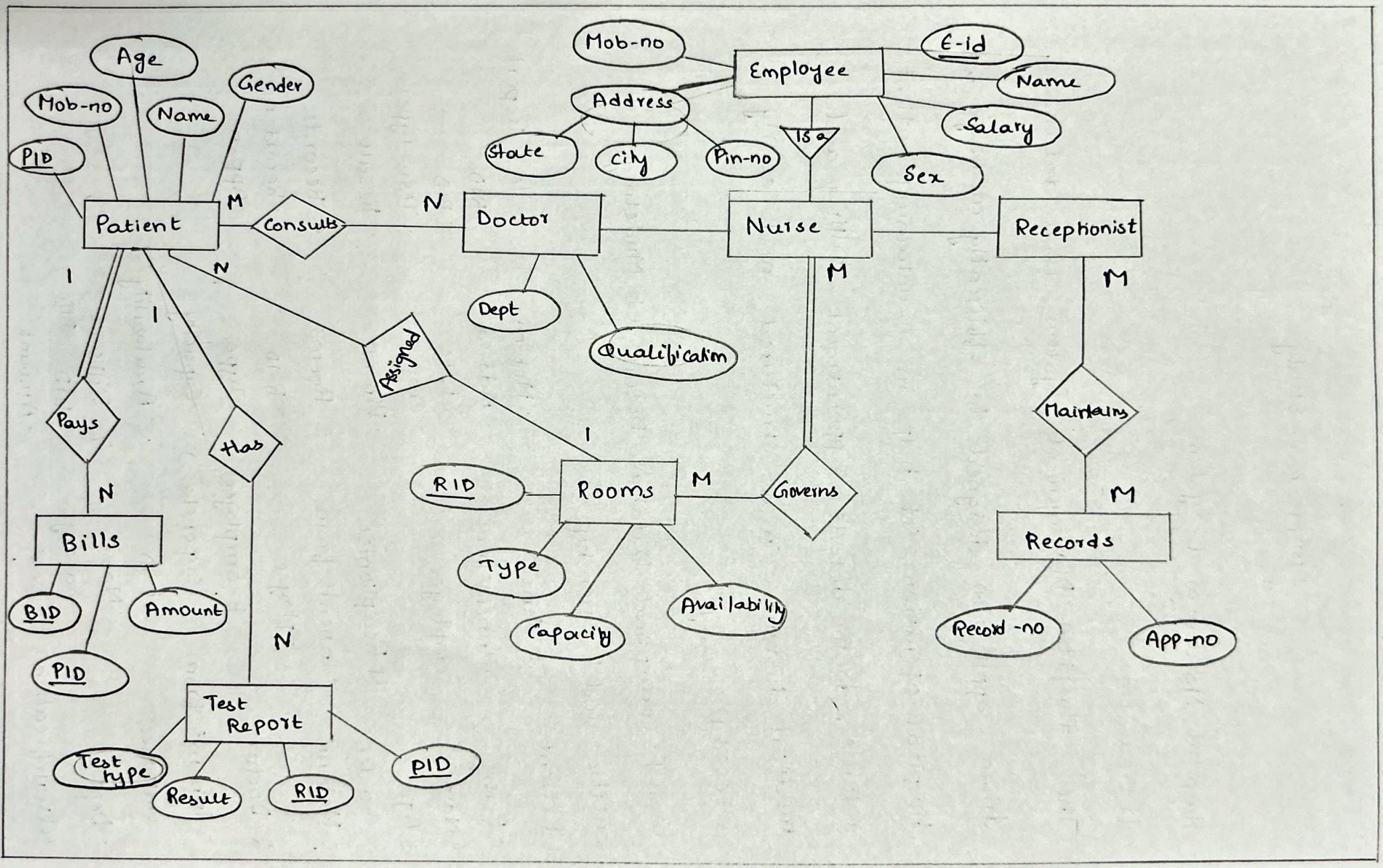
The Hospital Management System is a database driven application designed to efficiently manage all hospital operations. It helps in maintaining patient records, doctor details, appointments, billing and medical histories in a structured and easily accessible way.

Task 1: Conceptual Database Design - ER Modeling

Entities and Relationships:

1. Patient:
   - P-ID
   - Name
   - Age
   - Gender
   - Mob-no
2. Doctor:
   - Inherits from employee
   - Dept
   - Qualification
3. Nurse
   - Inherits from employee
4. Receptionist
   - Inherits from employee
5. Employee (Super class)
   - E-ID
   - Name
   - Salary
   - Sex
   - Mob-no
   - Address
   - State
   - City
   - Pin-no
6. Rooms
   - R-ID
   - Type
   - Capacity
   - Availability
7. Bills
   - BID
   - PID
   - Amount
8. Test Report
   - RID
   - PID
   - Test-Type
   - Result
9. Records
   - Record-no
   - App-no

ER- Diagram


Age
Mob-no
Name
Gender
PID
Patient
M
Consults
N
Doctor
Dept
Qualification
Mob-no
Employee
E-id
Name
Address
Salary
State
City
Pin-no
Is a
Sex
Nurse
Receptionist
M
M
1
N
1
Assigned
Pays
Has
1
N
N
Bills
BID
Amount
PID
Test Report
Test type
Result
RID
PID
RID
Rooms
M
Type
Capacity
Availability
Governs
Maintains
M
M
Records
Record -no
App-no

Task 2 - Logical Database Design - Schema Creation

- Converting the ER Diagram into relational tables:

1. EMPLOYEE (super class - parent for Doctor, Nurse, Receptionist)

Table: Employee

Attributes:

- E_ID (primary key)
- Name
- Salary
- Sex
- Mob_no
- Address
- State
- City
- Pin_no

2. DOCTOR (Subclass of Employee)

Table: Doctor

Attributes:

- E_ID (Primary key, Foreign key references Employee(E_ID))
- Dept
- Qualification

3. NURSE (Subclass of Employee)

Table: Nurse

Attributes:

- E_ID(Primary key, Foreign key references Employee(E_ID))

4. RECEPTIONIST (subclass of Employee)

Table: Receptionist

Attributes:

- E-ID (Primary key, Foreign key references Employee (E_ID))

5. PATIENT

Table: Patient

Attributes:

- P_ID (primary key)
- Name
- Age
- Gender
- Mob-no

6. ROOMS

Table: Rooms

Attributes:

- R_ID (primary key)
- Type
- Capacity
- Availability

7. BILLS

Table: Bills

Attributes:

- B_ID (primary key)
- P_ID (Foreign key references Patient (P_ID))
- Amount.

8. TEST_REPORT

Table: Test_Report

Attributes:

- R_ID (primary key)
- P_ID (Foreign key references Patient (P_ID))
- Test_Type
- Result

9. RECORDS

Table: Records

Attributes:

- Record_No (Primary key)
- App_No

Relationship Tables:

10. CONSULTS (B/w Patient & Doctor - M:N relationship)

Table: Consults

Attributes:

- P_ID (Foreign key references Patient(P_ID))
- E_ID (Foreign key references Doctor (E_ID))

Primary key: (P_ID, E_ID)

11. ASSIGNED (b/w Patient & Rooms - M:N relationship)

Table: Assigned

Attributes:

- P_ID (Foreign key references Patient (P_ID))
- R_ID (Foreign key references Rooms (R_ID))

Primary key: (P_ID, R_ID)

12. GOVERNS (b/w Nurse & Rooms - M:N relationship)

Table: Governs

Attributes:

- E_ID (Foreign key references Nurse (E_ID))
- R_ID (Foreign key references Rooms (R_ID))

Primary key: (E_ID, R_ID)

13. MAINTAINS (b/w Receptionist & Records - M:N relationship)

Table: Maintains

Attributes:

- E_ID (Foreign key references Receptionist (E_ID))
- Record_No (Foreign key references Records (Record_No))

Primary key: (E_ID, Record_No)

- Define primary keys, foreign keys and data types for each table.

1. EMPLOYEE Table

```
CREATE TABLE Employee(
    E_ID INT PRIMARY KEY,
    NAME VARCHAR(50) NOT NULL,
    SALARY DECIMAL(10,2),
    SEX CHAR(1),
    MOB_NO VARCHAR(15),
    ADDRESS VARCHAR(100),
    STATE VARCHAR(30),
    CITY VARCHAR(30),
    PIN_NO VARCHAR(30).
);
```

2. DOCTOR Table:

```
CREATE TABLE Doctor(
        E_ID INT PRIMARY KEY,
        Dept VARCHAR(40),
        Qualification VARCHAR(50),
        FOREIGN KEY(E_ID) REFERENCES Employee(E_ID)
);
```

3. NURSE Table:

```
CREATE TABLE Nurse(
        E_ID INT PRIMARY KEY,
      FOREIGN KEY(E_ID) REFERENCES Employee(E_ID)
);
```

4. RECEPTIONIST Table:

```
CREATE TABLE Receptionist(
        E_ID INT PRIMARY KEY,
      FOREIGN KEY(E_ID) REFERENCES Employee(E_ID)

);
```

5. PATIENT Table:

```
CREATE TABLE Patient(
        P_ID INT PRIMARY KEY,
        NAME VARCHAR(50) NOT NULL,
        AGE INT,
        GENDER CHAR(1),
        MOB_NO VARCHAR(15)
);
```

6. ROOMS Table:

```
CREATE TABLE Rooms(
        R_ID INT PRIMARY KEY,
        TYPE VARCHAR(30),
        CAPACITY INT,
        Availability VARCHAR(10)
);
```

7. BILLS Table:

```
CREATE TABLE BILLS(
        B_ID INT PRIMARY KEY,
        P_ID INT,
        Amount DECIMAL(10,2),
        FOREIGN KEY REFERENCES Patient(P_ID)
                P_ID
);
```

8. TEST_REPORT Table:

```
CREATE TABLE TEST_REPORT(
        R_ID INT PRIMARY KEY,
        P_ID INT,
        TEST_TYPE VARCHAR(40),
        RESULT VARCHAR(100),
        FOREIGN KEY (P_ID) REFERENCES Patient(P_ID)
);
```

9. RECORDS Table:

```
CREATE TABLE Records(
        Record-no INT PRIMARY KEY,
        App_no INT
);
```

Relationship Tables:

10. CONSULTS (Patient ↔ DOCTOR)

```
CREATE TABLE Consults(
        P_ID INT,
        E_ID INT,
      PRIMARY KEY(P_ID, E_ID),
      FOREIGN KEY(P_ID) REFERENCES Patient P_ID),
      FOREIGN KEY (E_ID) REFERENCES Doctor(E_ID)
);
```

11. ASSIGNED (Patient ↔ Rooms)

```
CREATE TABLE Assigned(
        P_ID INT,
        R_ID INT,
        PRIMARY KEY (P_ID, R_ID),
      FOREIGN KEY (P_ID) REFERENCES Patient(P_ID),
      FOREIGN KEY (R_ID) REFERENCES Rooms(R_ID)
);
```

12. GOVERNS (Nurse ↔ Rooms)

```
CREATE TABLE GOVERNS(
    E_ID INT,
    R_ID INT,
    PRIMARY KEY(E_ID, R_ID),
    FOREIGN KEY (E_ID) REFERENCES Nurse(E_ID),
    FOREIGN KEY (R_ID) REFERENCES Rooms (R_ID),
);
```

13. MAINTAINS (Receptionist ↔ Records)

```
CREATE TABLE Maintains (
        E_ID INT,
        Record_no INT,
        PRIMARY KEY (E_ID, Record_no)
    FOREIGN KEY(E_ID) REFERENCES Receptionist (E_ID),
    FOREIGN KEY (Record_no) REFERENCES Records (Record_no)
);
```

Task3 : Normalization

- Analyze the following table for redundancy & anomalies.

Example Table:

P_ID	P_Name	Age	Gender	Mob-no	Doctor-name	Dept
P101	Ravi	25	M	9876543210	Dr. Meena	Cardiology
P102	Sneha	30	F	9845632100	Dr. Arjun	Neurology
P103	Ravi	25	M	9876543210	Dr. Meena	Cardiology

Room_No	Room_Type	Bill_ID	Amount
R01	AC	B001	12000
R02	Non-AC	B002	8000
R03	AC	B003	13000

Identify Redundancy & Anomalies:

Redundancy:

- Patient "Ravi" details (name, age, gender, mobile) are repeated multiple times.

- Doctor and Room details are also duplicated for multiple patients.

Anomalies:-

- Insertion anomaly: Can't add a new doctor unless a patient is assigned.
- Update anomaly: If Dr. Meena's department changes, must update all rows.
- Deletion anomaly: Deleting a patient could delete doctor and room info.

- Identify Functional Dependencies (FDs)

From the table, we can see:

1. P_ID → P_Name, Age, Gender, Mob-no, Doctor_Name, Dept, Room_no, Room_Type, Bill_ID, Amount
2. Doctor_Name → Dept
3. Room_No → Room_Type
4. Bill_ID → Amount

- Identify Candidate Key

P_ID uniquely identifies each row →

Candidate Key = P_ID

- Decompose into 3NF

We create separate tables to remove redundancy.

(a) PATIENT Table:

```
CREATE TABLE Patient(
        P_ID VARCHAR(10) PRIMARY KEY,
        P_NAME VARCHAR(50)
        Age INT,
        Gender CHAR(1),
        Mob-No VARCHAR(15)
);
```

(b) DOCTOR Table:

```
CREATE TABLE Doctor(
        Doctor_Name VARCHAR(50) PRIMARY KEY,
        DEPT VARCHAR(40)
);
```

(c) ROOM Table:

```
CREATE TABLE Room(
        Room_No VARCHAR(10) PRIMARY KEY,
        Room_Type VARCHAR(20)
);
```

(d) BILL Table:

```
CREATE TABLE Bill(
        Bill_ID VARCHAR(10) PRIMARY KEY,
        AMOUNT Decimal(10,2)
);
```

(e) PATIENT_ASSIGNMENT Table :

```
CREATE TABLE  Patient_Assignment (
        P_ID VARCHAR(10),
        Doctor_Name VARCHAR(50),
        Room_No VARCHAR(10),
        BILL_ID VARCHAR(10),
        PRIMARY KEY (P_ID),
        FOREIGN KEY (P_ID) REFERENCES Patient(P_ID),
        FOREIGN KEY(Doctor_Name) REFERENCES Doctor(Doc_Name),
        FOREIGN KEY (Room_No) REFERENCES Room (Room_No),
        FOREIGN KEY (BILL_ID) REFERENCES Bill( BILL_ID)
);
```

- Insert Sample Data :

Patient

```
INSERT  INTO Patient VALUES
('P101', 'Ravi', 25, 'M', '9876543210'),
('P102', 'Sneha', 30, 'F', '9845632100'),
('P103', 'Ravi', 25, 'M', '9876543210');
```

Doctor

```
INSERT INTO Doctor VALUES
('Dr. Meena', 'Cardiology'),
('Dr. Arjun', 'Neurology');
```

Room

```
INSERT INTO Room VALUES
('R01', 'Ac'),
('R02', 'NonAc'),
('R03', 'Ac');
```

Bill

```
INSERT INTO Bill VALUES
('B001', 12000),
('B002', 8000),
('B003', 13000);
```

Patient-Assignment

```
INSERT INTO Patient_Assigment VALUES
('P101', 'Dr.Meena', 'RD1', 'B001'),
('P102', 'Dr.Arjun', 'R02', 'B002'),
('P103', 'Dr.Meena', 'R03', 'B003');
```

Final Normalized Structure (in 3NF)

Table	Primary key	Purpose
Patient	P_ID	Stores Patient details
Doctor	Doctor_Name	Stores Doctor Info
Room	Room_No	Stores Room details
Bill	Bill_ID	Stores Billing Info
Patient_Assignment	P_ID	Links patient with above

Task 4 : PL/SQL and Views

- Create a trigger to prevent entering a bill amount above ₹1,00,000 in the Bills table.

```
CREATE OR REPLACE TRIGGER c
BEFORE INSERT OR UPDATE ON Bills
FOR EACH ROW
BEGIN
    IF :NEW.Amount > 100000 THEN
        RAISE_APPLICATION_ERROR (-20001, 'Bill amount
        cannot exceed ₹1,00,000');
    END IF;
END;
/
```

This trigger runs before inserting or updating a bill record. If the entered Amount is greater than 100000, the trigger raises an error and prevents the operation.

- Create a view to display Patient Name, Doctor Name, Room Type, and Bill Amount.

```
CREATE OR REPLACE VIEW Patient_Details_View AS
SELECT
    p.Name AS Patient_Name,
    d.Doctor_Name,
    r.ROOM_Type,
    b.Amount AS Bill_Amount
FROM
    Patient P
    JOIN Patient_Assignment pa ON P.P_ID = Pa.P_ID
```

```
JOIN Doctor d ON pa.doctor_name=d.doctor_name
JOIN Room r ON Pa.Room_No = r. Room_No
JOIN Bill b ON Pa.Bill_ID = b.Bill_ID ;
```

This view joins multiple tables to provide a combined report of patient, doctor, room, and billing details - easy for hospital staff to query.

- Writing a query to Update a Bill Using the View

We can update the bill amount through the view using the following query.

```
UPDATE Patient_Details_View
SET Bill_Amount = 9500
WHERE Patient_Name = 'Sneha';
```

This updates Sneha's bill amount to ₹9,500 in the underlying Bills table through the view.