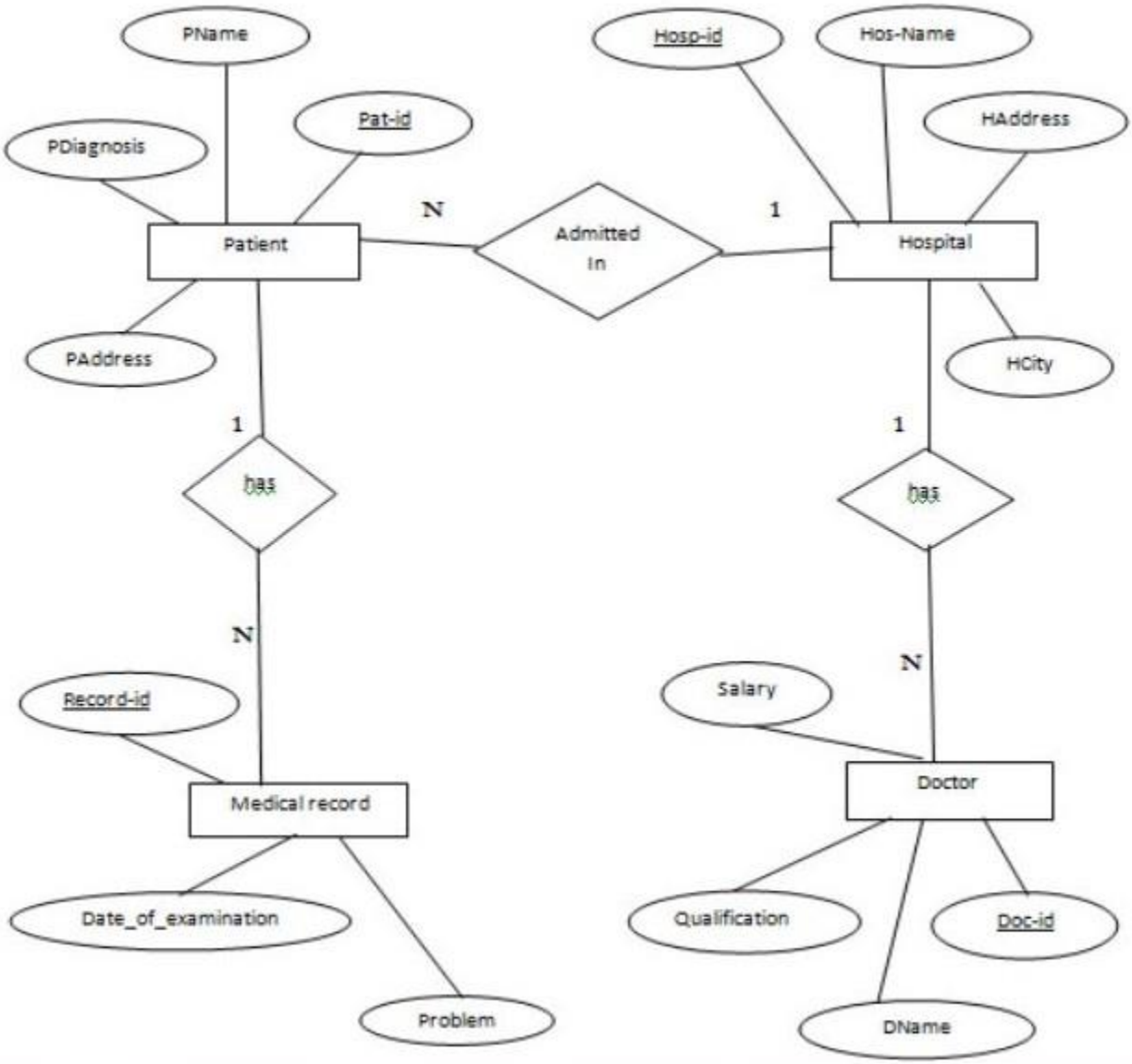
Assignment 1

1] Analyse a given business scenario and create an ER diagram that includes entities, relationships, attributes, and cardinality. Ensure that the diagram reflects proper normalization up to the third normal form.



Scenario: Hospital System

Scenario:

A hospital wants to design a database to manage its operations efficiently. The hospital consists of multiple departments, each headed by a department head. Each department offers various medical services. Doctors are assigned to departments based on their specialization. Each doctor may work in one or more departments, and each department may have multiple doctors.

Patients visit the hospital to receive medical treatment. Each patient is assigned a unique patient ID upon registration. Patients can visit one or more departments for consultations or treatments. During each visit, a patient is attended to by one or more doctors. Each visit is recorded with details such as date, time, department, and doctor's name.

Entities:

1. Patient
2. Hospital
3. Doctor
4. Medical Report

Relationships:

1. Each patient can have multiple visits. (One-to-Many)
2. Each visit is attended by one or more doctors, and each doctor can attend multiple visits. (Many-to-Many, resolved through an associative entity)
3. Each doctor works in one or more departments, and each department has multiple doctors. (Many-to-Many, resolved through an associative entity).

This ER diagram reflects proper normalization up to the third normal form. Each entity has been identified along with its attributes, relationships have been established, and associative entities have been introduced to resolve many-to-many relationships.

2] Design a database schema for a library system, including tables, fields, and constraints like NOT NULL, UNIQUE, and CHECK. Include primary and foreign keys to establish relationships between tables.

Table to store information about books

```
CREATE TABLE Books (
  BookID INT AUTO_INCREMENT PRIMARY KEY,
  Title VARCHAR(100) NOT NULL,
 Author VARCHAR(100) NOT NULL,
 ISBN VARCHAR(20) NOT NULL UNIQUE
);

CREATE TABLE Members (
  MemberID INT AUTO_INCREMENT PRIMARY KEY,
  FirstName VARCHAR(100) NOT NULL,
  LastName VARCHAR(100) NOT NULL,
  Email VARCHAR(200) NOT NULL UNIQUE,
  Phone VARCHAR(20),
);

CREATE TABLE Loan (
    LoanID INT AUTO_INCREMENT PRIMARY KEY,
```

```
 BookID INT NOT NULL,

 Borrower INT NOT NULL,

 FOREIGN KEY(Book ID) REFERENCES Books(BookID),

 FOREIGN KEY(BorrowerID) REFERENCES Borrowers(BorrowerID)

);
```

3] Explain the ACID properties of a transaction in your own words. Write SQL statements to simulate a transaction that includes locking and demonstrate different isolation levels to show concurrency control.

Properties:

1. **Atomicity:** This property ensures that a transaction is treated as a single unit of work. It means that either all the operations within a transaction are successfully completed and the changes are committed to the database, or none of the changes are applied if any operation fails. In other words, a transaction should be "all or nothing".
2. **Consistency:** Consistency ensures that the database remains in a valid state before and after the transaction. It means that transactions should follow all defined rules, constraints, and integrity constraints. The execution of a transaction should maintain the overall consistency of the database.
3. **Isolation:** Isolation ensures that the operations within a transaction are isolated from other concurrent transactions. Each transaction should appear to be executed in isolation, independently of other transactions, even if they are executed concurrently. Isolation prevents interference between transactions, ensuring that the data seen by one transaction remains consistent throughout its execution.
4. **Durability:** Durability guarantees that once a transaction is committed, the changes made by the transaction are permanent and will not be lost, even in the event of system failures such as power outages or crashe. Committed

changes should be stored permanently in the database and should be recoverable.

Transaction Simulation with Explicit Locking

```
-- Transaction 1

BEGIN TRANSACTION;


SELECT * FROM your_table WHERE id = 1 FOR UPDATE;


-- Update the row

UPDATE your_table SET column_name = new_value WHERE id = 1;


COMMIT;


-- Transaction 2

BEGIN TRANSACTION;


SELECT * FROM your_table WHERE id = 1 FOR UPDATE;


-- Update the row

UPDATE your_table SET column_name = new_value WHERE id = 1;
```

COMMIT;

Isolation level:

1. **Read Uncommitted:** This is the lowest isolation level. It allows transactions to read data that has been modified by other transactions but not yet committed. It can lead to dirty reads, where a transaction reads uncommitted data that may be rolled back later.
2. **Read Committed:** In this isolation level, transactions can only read data that has been committed by other transactions. It prevents dirty reads but may still allow non-repeatable reads and phantom reads.
3. **Repeatable Read:** This isolation level ensures that once a transaction reads a piece of data, it will always see the same value, even if other transactions modify or delete that data. However, it may still allow phantom reads, where new rows are inserted by other transactions.
4. **Serializable:** This is the highest isolation level. It provides the strictest guarantees by ensuring that transactions execute as if they were executed serially, one after the other. It prevents all anomalies, including dirty reads, non-repeatable reads, and phantom reads, by effectively serializing transactions.

4] Write SQL statements to CREATE a new database and tables that reflects the library schema you designed earlier. Use ALTER statements to modify the table structure and DROP statements to remove a redundant table.

```
CREATE TABLE Books (
  BookID INT AUTO_INCREMENT PRIMARY KEY,
  Title VARCHAR(100) NOT NULL,
 Author VARCHAR(100) NOT NULL,
```

```
 ISBN VARCHAR(20) NOT NULL UNIQUE
);

CREATE TABLE Members (
  MemberID INT AUTO_INCREMENT PRIMARY KEY,
  FirstName VARCHAR(100) NOT NULL,
  LastName VARCHAR(100) NOT NULL,
  Email VARCHAR(200) NOT NULL UNIQUE,
  Phone VARCHAR(20),
);

CREATE TABLE Loan (
    LoanID INT AUTO_INCREMENT PRIMARY KEY,
 BookID INT NOT NULL,
 Borrower INT NOT NULL,
 FOREIGN KEY(Book ID) REFERENCES Books(BookID),
 FOREIGN KEY(BorrowerID) REFERENCES Borrowers(BorrowerID)
);
```

Now, suppose we want to add a column genre to the Books table:

```
-- Add the genre column to the Books table
ALTER TABLE Books
```

ADD COLUMN genre VARCHAR(100) NOT NULL;

We also want to remove the phone_number column from the Members table:

Sql:

```
-- Remove the phone_number column from the Members table
ALTER TABLE Members
DROP COLUMN phone_number;
```

Suppose we want to remove the Loans table:

```
-- Drop the Loans table
DROP TABLE Loans;
```

5] Demonstrate the creation of an index on a table and discuss how it improves query performance. Use a DROP INDEX statement to remove the index and analyse the impact on query execution.

```
-- Create a table
CREATE TABLE Employee (
    employee_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
```

```
    department VARCHAR(50),
    salary DECIMAL(10, 2)
);

-- Insert sample data
INSERT INTO Employee (employee_id, first_name, last_name, department, salary)
VALUES
    (1, 'John', 'Doe', 'HR', 50000.00),
    (2, 'Jane', 'Smith', 'IT', 60000.00),
    (3, 'Mike', 'Johnson', 'Finance', 55000.00),
    (4, 'Emily', 'Brown', 'HR', 52000.00),
    (5, 'David', 'Jones', 'IT', 62000.00);

-- Create an index on the department column
CREATE INDEX idx_department ON Employee (department);
```

An index on the department column will improve the performance of queries that involve filtering or sorting by the department column. For example, if we frequently run queries to retrieve employees from a specific department, the index will allow the database to quickly locate the relevant rows without having to scan the entire table.

```
-- Drop the index on the department column
DROP INDEX idx_department ON Employee;
```

After dropping the index, queries that rely on filtering or sorting by the department column may experience degraded performance, especially if the table contains a large number of rows. Without the index, the database will need to perform a full table scan to locate the relevant rows, which can be slower compared to using an index.

6] Create a new database user with specific privileges using CREATE USER and GRANT commands. Then, write a script to REVOKE certain privileges and DROP the user.

Create a new user:

```
CREATE USER 'newuser'@'localhost' IDENTIFIED BY 'password';
```

Grant specific privileges to the user:

```
GRANT SELECT, INSERT, UPDATE ON dbname.* TO 'newuser'@'localhost';
```

Revoke certain privileges:

```
REVOKE INSERT, UPDATE ON dbname.* FROM 'newuser'@'localhost';
```

Drop the user:

```
DROP USER 'newuser'@'localhost';
```

7] Prepare a series of SQL statements to INSERT new records into the library tables, UPDATE existing records with new information, and DELETE records based on specific criteria. Include BULK INSERT operation to load data from an external source.

INSERT new records into the library tables

```
-- Insert a new book record

INSERT INTO Books (title, author, publication_year, ISBN, available_copies)

VALUES ('The Great Gatsby', 'F. Scott Fitzgerald', 1925, '9780743273565', 10);


-- Insert a new member record

INSERT INTO Members (name, email, phone_number, address)

VALUES ('John Smith', 'john@example.com', '123-456-7890', '123 Main St');


-- Insert a new loan record

INSERT INTO Loans (book_id, member_id, loan_date, return_date)

VALUES (1, 1, '2024-06-01', NULL);



--UPDATE Statement :

UPDATE books

SET genre = 'Classic Fiction'
```

```
WHERE title = 'To Kill a Mockingbird';


--DELETE Statement:

DELETE FROM books

WHERE published_year < 1950;




--BULK INSERT Statement:

BULK INSERT Books

FROM 'C:\path\to\new_books.csv'

WITH (

   FIELDTERMINATOR = ',',

   ROWTERMINATOR = '\n',

   FIRSTROW = 2 -- If the first row contains headers

);
```

These statements will update existing records and perform a bulk insert operation to load data from an external CSV file into the 'Books' table. Adjust file paths and table/column names as needed for your specific database setup.