Que 1. Analyze 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.

Ans.

Library Management System (3NF):

Scenario: Let's create an ER diagram for a library management system.

Entities:

1. **Book:** (BookID (PK), Title, Author, ISBN)
2. **Member:** (MemberID (PK), Name, Address, PhoneNumber)
3. **Loan:** (LoanID (PK), BookID (FK), MemberID (FK), LoanDate, ReturnDate)

Relationships:

1. A **Book** can be loaned to many **Members** (One-to-Many)
2. A **Member** can borrow many **Books** (One-to-Many)

Cardinality:

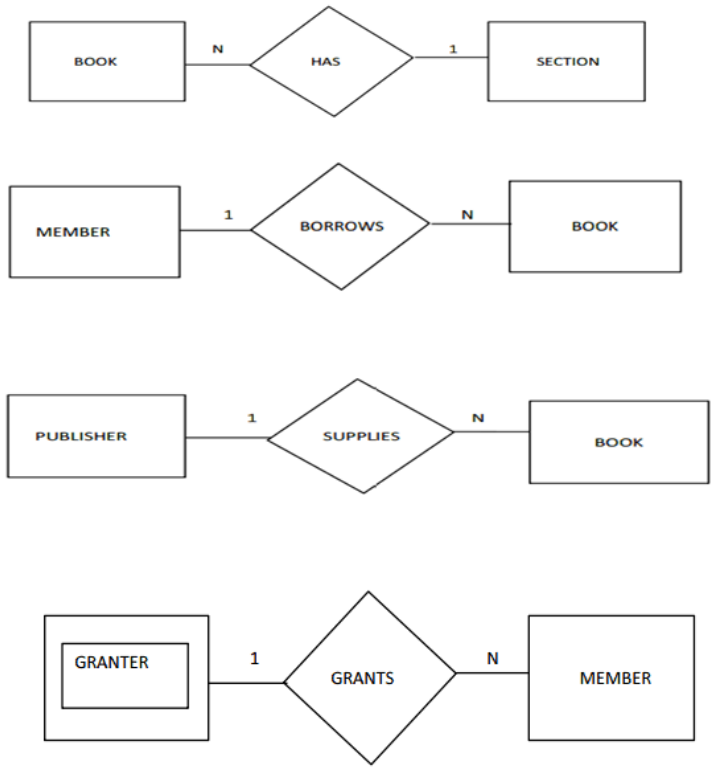
- **Book:** One book can be loaned to many members (1:N)
- **Member:** One member can borrow many books (1:N)
- **Loan:** One loan record is for one book loaned to one member (1:1)

Normalization:

This ER diagram is already in 3NF because:

- Each table has a primary key that uniquely identifies each record.
- There are no repeating groups within any table.
- All non-key attributes in the Loan table (LoanDate, ReturnDate) are fully dependent on the primary key (combination of BookID and MemberID), eliminating transitive dependencies.

3NF- form diagram:



Que 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.

Ans.

```
CREATE TABLE Authors (
    AuthorID INT AUTO_INCREMENT PRIMARY KEY,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL
);

CREATE TABLE Categories (
    CategoryID INT AUTO_INCREMENT PRIMARY KEY,
    CategoryName VARCHAR(100) NOT NULL UNIQUE
);
```

```
CREATE TABLE Books (
    BookID INT AUTO_INCREMENT PRIMARY KEY,
    Title VARCHAR(200) NOT NULL,
    ISBN VARCHAR(13) NOT NULL UNIQUE,
    PublishedYear YEAR NOT NULL CHECK (PublishedYear >= 1000 AND PublishedYear <=
YEAR(CURDATE())),
    AuthorID INT NOT NULL,
    CategoryID INT,
    CopiesAvailable INT NOT NULL CHECK (CopiesAvailable >= 0),
    FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID) ON DELETE CASCADE,
    FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID) ON DELETE SET NULL
);

CREATE TABLE Members (
    MemberID INT AUTO_INCREMENT PRIMARY KEY,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Email VARCHAR(100) NOT NULL UNIQUE,
    Phone VARCHAR(15),
    JoinDate DATE NOT NULL DEFAULT CURDATE()
);

CREATE TABLE Loans (
    LoanID INT AUTO_INCREMENT PRIMARY KEY,
    BookID INT NOT NULL,
    MemberID INT NOT NULL,
    LoanDate DATE NOT NULL DEFAULT CURDATE(),
    ReturnDate DATE,
    DueDate DATE NOT NULL CHECK (DueDate >= LoanDate),
```

```
    FOREIGN KEY (BookID) REFERENCES Books(BookID) ON DELETE CASCADE,
    FOREIGN KEY (MemberID) REFERENCES Members(MemberID) ON DELETE CASCADE
);
```

Que. 3

ACID properties of database transactions:

1. **Atomicity**: Ensures that a transaction is all-or-nothing. If any part of the transaction fails, the entire transaction is rolled back, leaving the database in its original state.
2. **Consistency**: Ensures that a transaction brings the database from one valid state to another, maintaining database integrity.
3. **Isolation**: Ensures that concurrent transactions occur independently without interfering with each other. The changes made by one transaction are not visible to other transactions until they are committed.
4. **Durability**: Ensures that once a transaction is committed, it remains so, even in the case of a system failure.

SQL Syntax:

```
CREATE TABLE Books (
    BookID INT PRIMARY KEY,
    Title VARCHAR(200) NOT NULL,
    CopiesAvailable INT NOT NULL CHECK (CopiesAvailable >= 0)
);

CREATE TABLE Loans (
    LoanID INT PRIMARY KEY AUTO_INCREMENT,
    BookID INT NOT NULL,
    MemberID INT NOT NULL,
    LoanDate DATE NOT NULL DEFAULT CURDATE(),
    FOREIGN KEY (BookID) REFERENCES Books(BookID)
);
```

Transaction with locking :

```
START TRANSACTION;

-- Lock the book record to prevent other transactions from modifying it
SELECT CopiesAvailable FROM Books WHERE BookID = 1 FOR UPDATE;

-- Check if the book is available
SET @available := (SELECT CopiesAvailable FROM Books WHERE BookID = 1);

IF @available > 0 THEN
   -- Decrease the number of available copies
   UPDATE Books SET CopiesAvailable = CopiesAvailable - 1 WHERE BookID = 1;

   -- Insert the loan record
   INSERT INTO Loans (BookID, MemberID) VALUES (1, 123);
END IF;

-- Commit the transaction
COMMIT;
```

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

Ans.

```
-- Create Database
CREATE DATABASE LibraryDB;
```

```
USE LibraryDB;

-- Create Tables
CREATE TABLE Authors (
    AuthorID INT AUTO_INCREMENT PRIMARY KEY,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL
);

CREATE TABLE Categories (
    CategoryID INT AUTO_INCREMENT PRIMARY KEY,
    CategoryName VARCHAR(100) NOT NULL UNIQUE
);

CREATE TABLE Books (
    BookID INT AUTO_INCREMENT PRIMARY KEY,
    Title VARCHAR(200) NOT NULL,
    ISBN VARCHAR(13) NOT NULL UNIQUE,
    PublishedYear YEAR NOT NULL CHECK (PublishedYear >= 1000 AND PublishedYear <=
YEAR(CURDATE())),
    AuthorID INT NOT NULL,
    CategoryID INT,
    CopiesAvailable INT NOT NULL CHECK (CopiesAvailable >= 0),
    FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID) ON DELETE CASCADE,
    FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID) ON DELETE SET NULL
);

CREATE TABLE Members (
    MemberID INT AUTO_INCREMENT PRIMARY KEY,
```

```
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Email VARCHAR(100) NOT NULL UNIQUE,
    Phone VARCHAR(15),
    JoinDate DATE NOT NULL DEFAULT CURDATE()
);

CREATE TABLE Loans (
    LoanID INT AUTO_INCREMENT PRIMARY KEY,
    BookID INT NOT NULL,
    MemberID INT NOT NULL,
    LoanDate DATE NOT NULL DEFAULT CURDATE(),
    ReturnDate DATE,
    DueDate DATE NOT NULL CHECK (DueDate >= LoanDate),
    FOREIGN KEY (BookID) REFERENCES Books(BookID) ON DELETE CASCADE,
    FOREIGN KEY (MemberID) REFERENCES Members(MemberID) ON DELETE CASCADE
);

-- Alter Table Structures
ALTER TABLE Authors
ADD MiddleName VARCHAR(50);

ALTER TABLE Members
MODIFY Phone VARCHAR(20);

-- Drop a Redundant Table
DROP TABLE IF EXISTS OldCategories;
```

Que. 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 analyze the impact on query execution.

Ans. By creating an index, we significantly improve query performance for operations that involve searching by Title. Removing the index causes the database to revert to slower full table scans, illustrating the importance of indexes for query optimization.

```
CREATE DATABASE LibraryDB;

USE LibraryDB;


CREATE TABLE Books (
   BookID INT AUTO_INCREMENT PRIMARY KEY,
   Title VARCHAR(200) NOT NULL,
   ISBN VARCHAR(13) NOT NULL UNIQUE,
   PublishedYear YEAR NOT NULL CHECK (PublishedYear >= 1000 AND PublishedYear <=
YEAR(CURDATE())),
   AuthorID INT NOT NULL,
   CategoryID INT,
   CopiesAvailable INT NOT NULL CHECK (CopiesAvailable >= 0),
   FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID) ON DELETE CASCADE,
   FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID) ON DELETE SET NULL
);
```

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

Ans.

```
-- Create the database user
CREATE USER 'library_user'@'localhost' IDENTIFIED BY 'secure_password';
```

```
-- Grant privileges to the user
GRANT SELECT, INSERT, UPDATE, DELETE ON LibraryDB.* TO 'library_user'@'localhost';

-- Apply the privileges
FLUSH PRIVILEGES;

-- Revoke certain privileges from the user
REVOKE INSERT, DELETE ON LibraryDB.* FROM 'library_user'@'localhost';

-- Drop the user
DROP USER 'library_user'@'localhost';
```

Que 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 operations to load data from an external source.

Insert Statement:

```
INSERT INTO Books (Title, ISBN, PublishedYear, AuthorID, CategoryID, CopiesAvailable)
VALUES ('The Great Gatsby', '9780743273565', 1925, 1, 1, 3);

INSERT INTO Books (Title, ISBN, PublishedYear, AuthorID, CategoryID, CopiesAvailable)
VALUES ('To Kill a Mockingbird', '9780060935467', 1960, 2, 2, 5);

INSERT INTO Books (Title, ISBN, PublishedYear, AuthorID, CategoryID, CopiesAvailable)
VALUES ('1984', '9780451524935', 1949, 3, 3, 4);

INSERT INTO Books (Title, ISBN, PublishedYear, AuthorID, CategoryID, CopiesAvailable)
VALUES ('The Great Gatsby', '9780743273565', 1925, 1, 1, 3);
```

```
INSERT INTO Books (Title, ISBN, PublishedYear, AuthorID, CategoryID, CopiesAvailable)
VALUES ('To Kill a Mockingbird', '9780060935467', 1960, 2, 2, 5);

INSERT INTO Books (Title, ISBN, PublishedYear, AuthorID, CategoryID, CopiesAvailable)
VALUES ('1984', '9780451524935', 1949, 3, 3, 4);
```

Update Book:

```
UPDATE Books
SET CopiesAvailable = CopiesAvailable + 1
WHERE BookID = 1;
```

Delete Book:

```
DELETE FROM Books
WHERE Title = '1984';
```