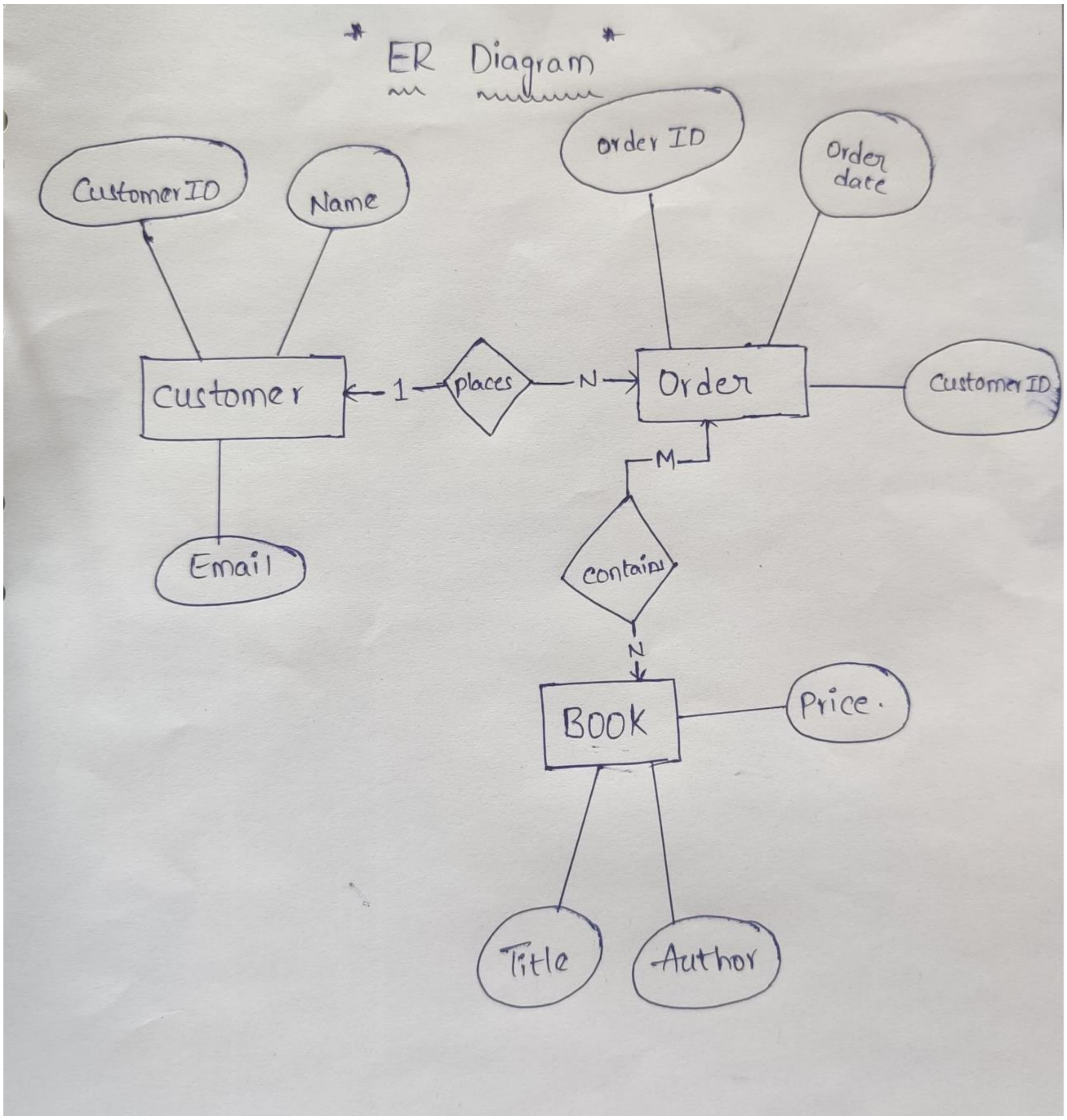
## Assignment 1:

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.

**ER Diagram:**



**Explanation:**

- The Customer entity has attributes CustomerID, Name, and Email.
- The Order entity has attributes OrderID, OrderDate, and CustomerID (which is a foreign key referencing Customer).
- The Book entity has attributes Title, Author, and Price.
- The Customer places an Order (1:N relationship).
- The Order contains Book (M:N relationship, which would typically involve an associative entity like OrderDetails).

## Assignment 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.

## Answer:

- **Atomicity**: This property ensures that all the operations within a transaction are treated as a single unit. Either all of them are executed successfully, or none are. It's like saying, "Do everything or do nothing."
- **Consistency**: Consistency ensures that a transaction can only bring the database from one valid state to another, maintaining the database's predefined rules, such as unique keys, foreign keys, and constraints.
- **Isolation**: Isolation determines how transaction integrity is visible to other users and systems. A transaction should appear as though it is the only operation being executed in the system.
- **Durability**: Once a transaction has been committed, it will remain so, even in the event of a system failure. This means the changes made by the transaction are permanent and must be stored in non-volatile memory.

**SQL Statements:**

```
-- Start a transaction with explicit locking

BEGIN TRANSACTION;


-- Assume we have a table `accounts` with columns `id`, `user_name`, and `balance`

-- Let's lock the account with id 1 for update
```

```
SELECT * FROM accounts WHERE id = 1 FOR UPDATE;


-- Perform some operations, like transferring money from one account to another

UPDATE accounts SET balance = balance - 100 WHERE id = 1;

UPDATE accounts SET balance = balance + 100 WHERE id = 2;


-- End the transaction

COMMIT;


-- To demonstrate different isolation levels, we can set the isolation level at the beginning of the transaction

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

-- Other levels include READ COMMITTED, REPEATABLE READ, and SERIALIZABLE
```

## Assignment 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.

**Program:**

```
create database library_system;

use library_system;


 -- Table for storing book details


CREATE TABLE Books (

    BookID INT PRIMARY KEY,

    Title VARCHAR(255) NOT NULL,
    Author VARCHAR(255) NOT NULL,
    ISBN VARCHAR(13) UNIQUE NOT NULL,
    PublicationYear YEAR,
    Genre VARCHAR(100),
    CHECK (PublicationYear > 1800)
);

-- Table for storing member details
CREATE TABLE Members (
    MemberID INT PRIMARY KEY,
    FirstName VARCHAR(255) NOT NULL,
    LastName VARCHAR(255) NOT NULL,
    Email VARCHAR(255) UNIQUE NOT NULL,
    JoinDate DATE NOT NULL,
);

-- Table for storing book loans

CREATE TABLE BookLoans (
    LoanID INT PRIMARY KEY,
    BookID INT,
    MemberID INT,
    IssueDate DATE NOT NULL,
    DueDate DATE NOT NULL,
    ReturnDate DATE,
    FOREIGN KEY (BookID) REFERENCES Books(BookID),
    FOREIGN KEY (MemberID) REFERENCES Members(MemberID),
    CHECK (IssueDate <= DueDate)
);
```

## Assignment 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.

**Program:**

```
create database library_system;
use library_system;
CREATE TABLE Books (
    BookID INT PRIMARY KEY,
    Title VARCHAR(255) NOT NULL,
    Author VARCHAR(255) NOT NULL,
    ISBN VARCHAR(13) UNIQUE NOT NULL,
    PublicationYear YEAR,
    Genre VARCHAR(100),
    CHECK (PublicationYear > 1800)
);
CREATE TABLE Members (
    MemberID INT PRIMARY KEY,
    FirstName VARCHAR(255) NOT NULL,
    LastName VARCHAR(255) NOT NULL,
    Email VARCHAR(255) UNIQUE NOT NULL,
    JoinDate DATE NOT NULL,
);

CREATE TABLE BookLoans (

    LoanID INT PRIMARY KEY,

    BookID INT,

    MemberID INT,

    IssueDate DATE NOT NULL,

    DueDate DATE NOT NULL,

    ReturnDate DATE,

    FOREIGN KEY (BookID) REFERENCES Books(BookID),

    FOREIGN KEY (MemberID) REFERENCES Members(MemberID),

    CHECK (IssueDate <= DueDate)

);


-- Alter the 'Books' table to add a new column for 'Publisher'

ALTER TABLE Books

ADD Publisher VARCHAR(255);


-- Drop the 'Genre' column from the 'Books' table as it is redundant

ALTER TABLE Books

DROP COLUMN Genre;


-- Remove the 'BookLoans' table if it's no longer needed

DROP TABLE BookLoans;
```

## Assignment 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.

**Explanation:**

Let’s take the Books table from the library database schema as an example. We’ll create an index on the Author column, which is a common field for search queries.

**-- Create an index on the 'Author' column of the 'Books' table**

**CREATE INDEX idx_author ON Books(Author);**

Creating an index on the Author column allows the database to quickly locate the rows associated with a particular author without scanning the entire table. This is similar to an index in a book, which helps you find information quickly without reading every page. When a query searches for books by a specific author, the database uses the index to efficiently locate all books by that author.

Now, let’s discuss the impact of removing this index:

**-- Drop the index 'idx_author' from the 'Books' table**

**DROP INDEX idx_author ON Books;**

Dropping the index means that the database will no longer have a quick reference for the Author column. Consequently, query performance can degrade, especially for large tables, because the database must perform a full table scan to find rows matching the query criteria, which is much slower than using an index.

In summary, an index can significantly improve query performance by providing a fast path to the data rows in a table based on the indexed columns. However, indexes also have drawbacks:

They consume additional storage space.

They can slow down write operations like INSERT, UPDATE, and DELETE, as the index must be updated in addition to the table data.

## Assignment 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."

**Program:**

```
-- INSERT statements for each table
-- Inserting a new book record
INSERT INTO Books (BookID, Title, Author, ISBN, PublicationYear)
VALUES (1, 'The Great Gatsby', 'F. Scott Fitzgerald', '1234567890123', 1925);

-- Inserting a new member record
INSERT INTO Members (MemberID, FirstName, LastName, Email, JoinDate)
VALUES (1, 'John', 'Doe', 'john.doe@example.com', '2024-05-14');

-- Inserting a new book loan record
INSERT INTO BookLoans (LoanID, BookID, MemberID, IssueDate, DueDate)
VALUES (1, 1, 1, '2024-05-14', '2024-06-14');

-- UPDATE statements
-- Updating a book's title
UPDATE Books
SET Title = 'The Great Gatsby: Revised Edition'
```

```
WHERE BookID = 1;

-- Updating a member's email
UPDATE Members
SET Email = 'john.newemail@example.com'
WHERE MemberID = 1;

-- DELETE statements
-- Deleting a book loan record
DELETE FROM BookLoans
WHERE LoanID = 1;

-- BULK INSERT operation (the file path and format may vary based on the system)
-- Bulk insert book records from an external CSV file
BULK INSERT Books
FROM 'C:\\path_to_your_file\\books.csv'
WITH
(
   FIELDTERMINATOR = ',',  -- CSV field delimiter
   ROWTERMINATOR = '\\n',   -- CSV row delimiter
   FIRSTROW = 2            -- If the first row contains column headers
);
```

## Assignment 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**.**

**Program:**

```
-- Create a new user 'library_user' with a password
CREATE USER 'library_user'@'localhost' IDENTIFIED BY 'Inthu@1234';


-- Grant SELECT, INSERT, and UPDATE privileges on the 'LibrarySystem' database to
the new user
GRANT SELECT, INSERT, UPDATE ON LibrarySystem.* TO
'library_user'@'localhost';


-- Revoke UPDATE privilege from the user
REVOKE UPDATE ON LibrarySystem.* FROM 'library_user'@'localhost';


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