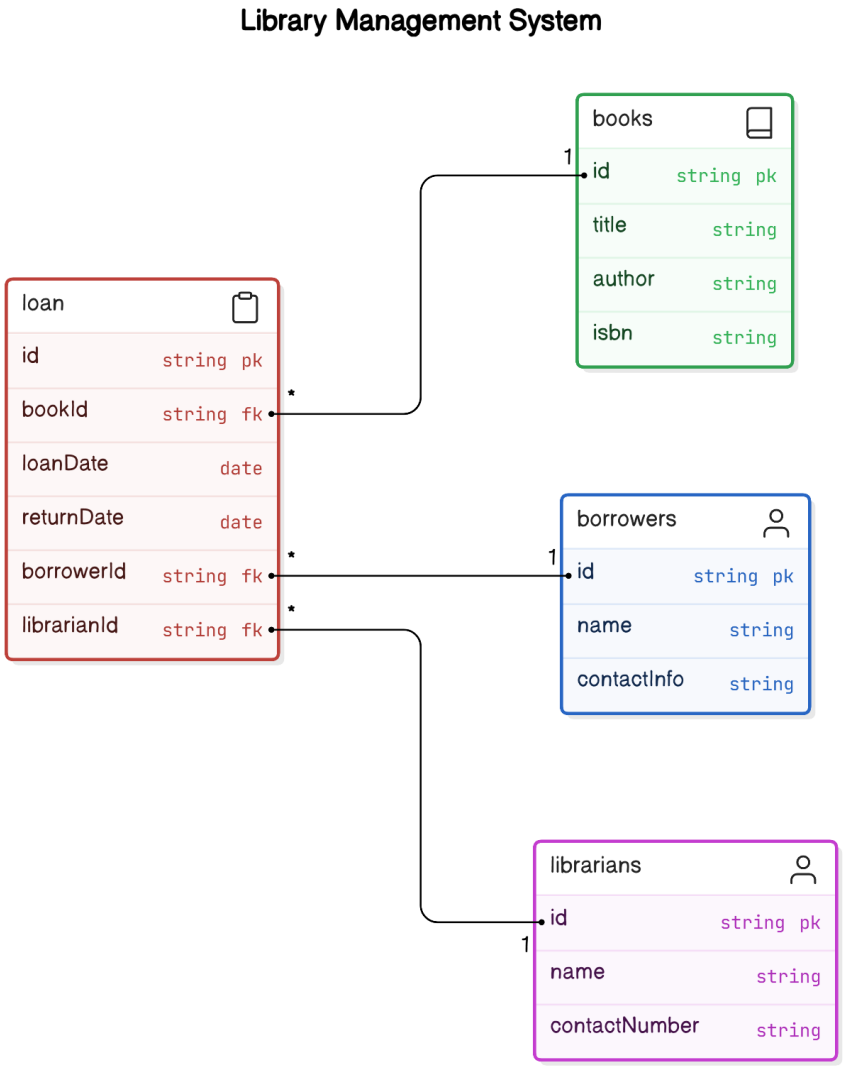
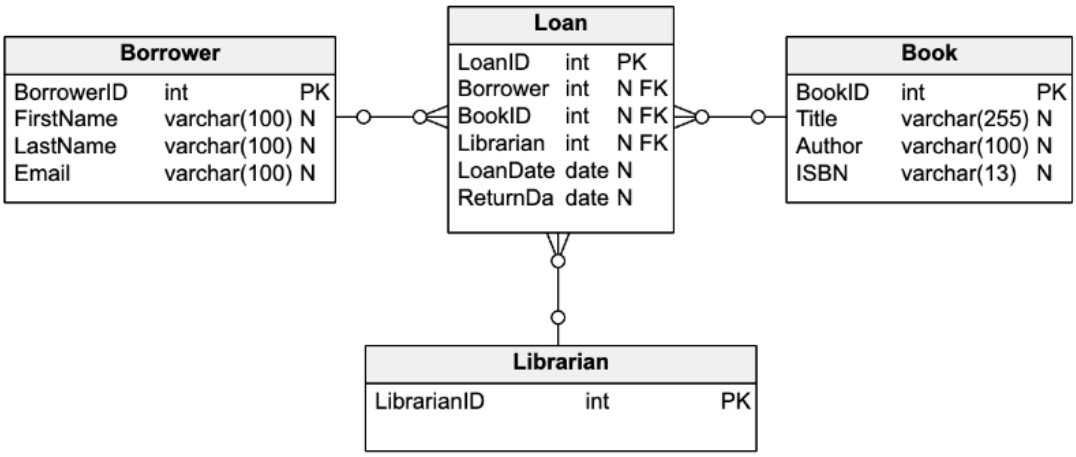
Assignment 1: Analyze a given business scenario and create an ER diagram that includes entities, relationships, attributes, and cardinality. Ensure that the diagram reflects proper normalization up to the third normal form.



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.



Entities: Book, Borrower, Loan, Librarian.
Relationships: Borrower 'borrows' Book, Loan 'recorded by' Librarian.
In this schema:

- The Borrower table represents the library patrons who borrow books.
- The Librarian table represents the librarians who record the book loans.
- The Book table represents the books that can be borrowed.
- The Loan table represents all the loan (i.e. borrowing) transactions. Each loan is associated with a borrower (who borrowed the book), a book (that was borrowed), and a librarian these are represented by the foreign keys BorrowerID, BookID, and librarianID The LoanDate and ReturnDate fields represent the date when the book was borrowed and the date when it was returned.

Description: This diagram depicts a library management system. Borrowers (library patrons) can borrow books. Each loan transaction, which pairs a borrower with a book, is recorded by a librarian.

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.

- Atomicity: Atomicity ensures that a transaction is treated as a single unit of work, which means that either all of its operations are completed successfully, or none of them are. In other words, if any part of the transaction fails, the entire transaction is rolled back, ensuring that the database remains in a consistent state.
- Consistency: Consistency ensures that the database remains in a valid state before and after the transaction. This means that any constraints, rules, or relationships defined in the database schema are maintained throughout the transaction. Transactions should not violate integrity constraints, ensuring that the data is always valid.
- Isolation: Isolation ensures that the execution of transactions concurrently does not interfere with each other. Each transaction should operate independently of other transactions, even if they are executing simultaneously. Isolation levels define the degree to which transactions are isolated from each other, preventing phenomena like dirty reads, non-repeatable reads, and phantom reads.
- Durability: Durability guarantees that once a transaction is committed, its changes are permanently saved in the database and will not be lost, even in the event of a system failure. This ensures that the data remains consistent over time and can be recovered in case of failures.

Inventory Example

```
BEGIN TRANSACTION;
LOCK TABLE inventory;
UPDATE inventory SET quantity = quantity - 1 WHERE item_id = 123;
-- Other operations within the transaction
COMMIT;
```

SETTING TRANSACTION LEVEL

Set isolation level to Read Uncommitted

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
SELECT quantity FROM inventory WHERE item_id = 123;
```

Set isolation level to Read Committed

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SELECT quantity FROM inventory WHERE item_id = 123;
```

Set isolation level to Repeatable Read

```
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
SELECT quantity FROM inventory WHERE item_id = 123;
```

Set isolation level to Serializable

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SELECT quantity FROM inventory WHERE item_id = 123;
```

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.

```
CREATE DATABASE IF NOT EXISTS LibraryDB;
USE LibraryDB;
CREATE TABLE Borrower (
   BorrowerID INT PRIMARY KEY AUTO_INCREMENT,
   FirstName VARCHAR(50),
   LastName VARCHAR(50),
   Email VARCHAR(255)
);
```

```
mysql> desc borrower;
+----------------+--------------+------+-----+---------+----------------+
| Field          | Type         | Null | Key | Default | Extra          |
+----------------+--------------+------+-----+---------+----------------+
| BorrowerID     | int          | NO   | PRI | NULL    | auto_increment |
| FirstName      | varchar(50)  | YES  |     | NULL    |                |
| LastName       | varchar(50)  | YES  |     | NULL    |                |
| Email          | varchar(255) | YES  |     | NULL    |                |
| MembershipType | varchar(50)  | YES  |     | NULL    |                |
+----------------+--------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

```
CREATE TABLE Librarian (
   LibrarianID INT PRIMARY KEY AUTO_INCREMENT
);
```

```
mysql> desc librarian;
+-------------+------+------+-----+---------+----------------+
| Field       | Type | Null | Key | Default | Extra          |
+-------------+------+------+-----+---------+----------------+
| LibrarianID | int  | NO   | PRI | NULL    | auto_increment |
+-------------+------+------+-----+---------+----------------+
1 row in set (0.00 sec)
```

```
CREATE TABLE Book (
    BookID INT PRIMARY KEY AUTO_INCREMENT,
    Title VARCHAR(255),
    Author VARCHAR(255),
    ISBN VARCHAR(13)
);
```

```
mysql> desc book;
+--------+--------------+------+-----+---------+----------------+
| Field  | Type         | Null | Key | Default | Extra          |
+--------+--------------+------+-----+---------+----------------+
| BookID | int          | NO   | PRI | NULL    | auto_increment |
| Title  | varchar(255) | YES  |     | NULL    |                |
| Author | varchar(255) | YES  |     | NULL    |                |
| ISBN   | varchar(13)  | YES  |     | NULL    |                |
+--------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

```
CREATE TABLE Loan (
    LoanID INT PRIMARY KEY AUTO_INCREMENT,
    BorrowerID INT,
    BookID INT,
    LibrarianID INT,
    LoanDate DATE,
    ReturnDate DATE,
    FOREIGN KEY (BorrowerID) REFERENCES Borrower(BorrowerID),
    FOREIGN KEY (BookID) REFERENCES Book(BookID),
    FOREIGN KEY (LibrarianID) REFERENCES Librarian(LibrarianID)
);
```

```
mysql> desc loan;
+-------------+------+------+-----+---------+----------------+
| Field       | Type | Null | Key | Default | Extra          |
+-------------+------+------+-----+---------+----------------+
| LoanID      | int  | NO   | PRI | NULL    | auto_increment |
| BorrowerID  | int  | YES  | MUL | NULL    |                |
| BookID      | int  | YES  | MUL | NULL    |                |
| LibrarianID | int  | YES  | MUL | NULL    |                |
| LoanDate    | date | YES  |     | NULL    |                |
| ReturnDate  | date | YES  |     | NULL    |                |
+-------------+------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

ALTER TABLE Borrower
ADD COLUMN MembershipType VARCHAR(50);
DROP TABLE IF EXISTS Publisher;

```
mysql> desc borrower;
+----------------+--------------+------+-----+---------+----------------+
| Field          | Type         | Null | Key | Default | Extra          |
+----------------+--------------+------+-----+---------+----------------+
| BorrowerID     | int          | NO   | PRI | NULL    | auto_increment |
| FirstName      | varchar(50)  | YES  |     | NULL    |                |
| LastName       | varchar(50)  | YES  |     | NULL    |                |
| Email          | varchar(255) | YES  |     | NULL    |                |
| MembershipType | varchar(50)  | YES  |     | NULL    |                |
+----------------+--------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

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.

Creating table Book

```
CREATE TABLE Book (
    BookID INT PRIMARY KEY,
    Title VARCHAR(255),
    Author VARCHAR(255),
    ISBN VARCHAR(13)
);
```

Index Creation
CREATE INDEX idx_title ON Book (Title);

Drop Index
DROP INDEX IF EXISTS idx_title ON Book;

How Index improves query performance

- Faster Data Retrieval: When a query filters, sorts, or joins data based on the indexed column(s), the database engine can use the index to locate the relevant rows more efficiently. Instead of scanning the entire table, it can perform an index seek or scan, which is generally faster.
- Reduced Disk I/O: Indexes store a sorted copy of the indexed column(s), which reduces the amount of disk I/O required for query processing. This is particularly beneficial for large tables, as it minimizes the need to read data from disk.

- Optimized Sorting and Join Operations: Indexes can improve the performance of sorting and join operations by providing an ordered sequence of values. This can lead to fewer disk accesses and CPU cycles required to process the query.

Impact on query execution after DROP INDEX

After dropping the index, queries that relied on the index for efficient data retrieval may experience degraded performance. The database engine may need to resort to full table scans or other less efficient access methods, which can lead to slower query execution times, especially for queries involving filtering, sorting, or joining based on the "Title" column.

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.

Create a new database user with specific privileges using the CREATE USER and GRANT commands:

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

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

```
mysql> GRANT SELECT, INSERT, UPDATE ON testdb.* TO 'newuser'@'localhost';
Query OK, 0 rows affected (0.02 sec)

mysql>
```

script to REVOKE certain privileges:

REVOKE INSERT ON your_database.* FROM 'newuser'@'localhost';

```
mysql> REVOKE INSERT ON testdb.* FROM 'newuser'@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

Drop the user:

DROP USER 'newuser'@'localhost';

```
mysql> DROP USER 'newuser'@'localhost';
Query OK, 0 rows affected (0.01 sec)
```

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.

```
INSERT INTO Borrower (BorrowerID, FirstName, LastName, Email)
VALUES (1, 'John', 'Doe', 'john.doe@example.com');

INSERT INTO Book (BookID, Title, Author, ISBN)
VALUES (1, 'The Great Gatsby', 'F. Scott Fitzgerald', '9780743273565');

INSERT INTO Loan (LoanID, BorrowerID, BookID, LibrarianID, LoanDate, ReturnDate)
VALUES (1, 1, 1, 1, '2024-05-21', NULL);
```

UPDATE existing records with new information:

```
UPDATE Borrower
SET Email = 'johndoe@example.com'
WHERE BorrowerID = 1;
```

DELETE records based on specific criteria:

```
DELETE FROM Loan
WHERE ReturnDate IS NULL;
```

BULK INSERT operations to load data from an external source:

```
SQL*Loader:
LOAD DATA
INFILE '/path/to/books.csv'
INTO TABLE books
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
(title, author, publication_year, isbn);
```

Note: The above statement assumes that the CSV file has headers. If it doesn't, you can use the TRAILING NULLCOLS option to specify the number of columns in the file.

Step 5 : Insert new records into the authors table:

```
INSERT INTO authors (name, birth_year)
VALUES (‘Sidney Sheldon’, 1980),
      (‘Eiichiro Oda’, 1990);
```

Step 6 : Update existing records in the authors table:

```
UPDATE authors
SET name = ‘Tom Clancy’
```

WHERE name = ‘Sidney Sheldon’;

Step 7 : Delete records from the authors table based on specific criteria:

```
DELETE FROM authors
WHERE birth_year > 1990;
```

Step 8: Bulk insert new records into the authors table from an external source using SQL*Loader:

```
LOAD DATA
INFILE '/path/to/authors.csv'
INTO TABLE authors
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
(name, birth_year);
```

----------End----------