Day - 9 Assignment (RDBMS and SQL) **Ankit Kumar Keshri**

Assignment 3: 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.

```
CREATE TABLE Authors (
    author_id INT AUTO_INCREMENT PRIMARY KEY,
    author_name VARCHAR(100) NOT NULL
);

-- Table: Books
CREATE TABLE Books (
    book_id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    isbn VARCHAR(13) UNIQUE,
    author_id INT,
    genre VARCHAR(100),
    publication_year INT,
    copies_available INT CHECK(copies_available >= 0),
    FOREIGN KEY (author_id) REFERENCES Authors(author_id)
);

-- Table: Members
CREATE TABLE Members (
    member_id INT AUTO_INCREMENT PRIMARY KEY,
```

```
    member_name VARCHAR(100) NOT NULL,
    email VARCHAR(255) UNIQUE,
    phone_number VARCHAR(20)
);

-- Table: Loans
CREATE TABLE Loans (
    loan_id INT AUTO_INCREMENT PRIMARY KEY,
    book_id INT,
    member_id INT,
    loan_date DATE NOT NULL,
    return_date DATE,
    FOREIGN KEY (book_id) REFERENCES Books(book_id),
    FOREIGN KEY (member_id) REFERENCES Members(member_id)
);

-- Table: Reservations
CREATE TABLE Reservations (
    reservation_id INT AUTO_INCREMENT PRIMARY KEY,
    book_id INT,
    member_id INT,
    reservation_date DATE NOT NULL,
    pickup_date DATE,
    status ENUM('Pending', 'Completed', 'Cancelled') DEFAULT 'Pending',
    FOREIGN KEY (book_id) REFERENCES Books(book_id),
    FOREIGN KEY (member_id) REFERENCES Members(member_id)
);
```

Assignment 4: 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 .

The ACID properties are a set of four characteristics that ensure the reliability and consistency of transactions in a database system:

Atomicity: Atomicity ensures that a transaction is treated as a single unit of work. This means that either all the operations within the transaction are successfully completed and committed to the database, or none of them are. There's no partial execution or incomplete state left in the database.

Consistency: Consistency ensures that the database remains in a valid state before and after the transaction. This means that all constraints, such as foreign key constraints or unique constraints, are enforced, and the integrity of the data is maintained. Transactions should transform the database from one valid state to another valid state.

Isolation: Isolation ensures that the concurrent execution of transactions does not result in any data inconsistency. Each transaction should appear to be executing in isolation, without being affected by other concurrently executing transactions. Isolation levels determine howtransactions interact with each other, and they range from allowing maximum concurrency (but potentially introducing anomalies) to strictly serializing transactions.

Durability: Durability ensures that once a transaction is committed, its changes are permanent and survive system failures. Even in the event of a crash or power failure, the changes made by committed transactions should not be lost.

let's simulate a transaction in SQL using locking and demonstrate different isolation levels:

```
BEGIN TRANSACTION;

UPDATE Accounts SET balance = balance - 100 WHERE account_id = 123;
UPDATE Accounts SET balance = balance + 100 WHERE account_id = 456;

COMMIT TRANSACTION;
```

This SQL code initiates a transaction that transfers $100 from account 123 to account 456. The BEGIN TRANSACTION statement marks the beginning of the transaction, and the COMMIT TRANSACTION statement commits the transaction, making the changes permanent.

To demonstrate different isolation levels, you can set the isolation level before starting the transaction:

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
BEGIN TRANSACTION;
COMMIT TRANSACTION;
```

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.

```
CREATE TABLE Employees (

 employee_id INT PRIMARY KEY,
 first_name VARCHAR(50),
 last_name VARCHAR(50),
 department VARCHAR(50),
 salary DECIMAL(10, 2)
);

SELECT * FROM Employees WHERE department = 'IT';

CREATE INDEX idx_department ON Employees(department);
```

After creating the index, the query execution time for retrieving employees based on their department should be faster. This is because the index allows the database engine to quickly locate the rows that match the specified department value without scanning the entire table.

```
DROP INDEX idx_department ON Employees;
```

In summary, while indexes can significantly improve query performance, they should be used strategically, considering factors such as the frequency of queries and the impact on data modification operations.

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 USER new_user WITH PASSWORD 'password';

GRANT SELECT, INSERT, UPDATE ON table_name TO new_user;
GRANT USAGE ON SCHEMA schema_name TO new_user;
REVOKE UPDATE ON table_name FROM new_user;

DROP USER new_user;
```

Explanation:
CREATE USER is used to create a new database user.

GRANT is used to grant specific privileges to the user. In this example, the user is granted SELECT, INSERT, and UPDATE privileges on a specific table (table_name), and USAGE privilege on a schema (schema_name).

REVOKE is used to revoke specific privileges from the user. In this example, the UPDATE privilege on the table is revoked from the user.

DROP USER is used to drop the user from the database once their privileges are no longer needed.

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.

Assuming we have the following tables: Books, Authors, Members, and Loans.

```
INSERT INTO Books (title, author_id, genre, publication_year, isbn)
VALUES ('The Great Gatsby', 1, 'Fiction', 1925, '9780743273565');

INSERT INTO Authors (author_name)
VALUES ('F. Scott Fitzgerald');

INSERT INTO Members (member_name, email, phone)
VALUES ('John Doe', 'john@example.com', '123-456-7890');

INSERT INTO Loans (book_id, member_id, loan_date, return_date, returned)
VALUES (1, 1, '2024-05-10', NULL, FALSE);

UPDATE Books
SET genre = 'Classic'
WHERE title = 'The Great Gatsby';

UPDATE Members
SET phone = '987-654-3210'
WHERE member_name = 'John Doe';


DELETE FROM Members
WHERE member_name = 'John Doe';
```

Suppose you have a CSV file named books.csv containing book data with columns title, author_id, genre, publication_year, and isbn. You can use a BULK INSERT operation to load this data into the Books table

```
COPY Books (title, author_id, genre, publication_year, isbn)
```

```
FROM '/path/to/books.csv'

DELIMITER ','
CSV HEADER;
```

These SQL statements demonstrate how to perform basic data manipulation operations in a library database, including inserting new records, updating existing records, deleting recordsbased on specific criteria, and bulk inserting data from an external source.