ACID Properties of a Transaction

- **Atomicity:** This ensures that a series of operations within a transaction are treated as a single unit. Either all operations are performed successfully, or none of them are. If any part of the transaction fails, the entire transaction is rolled back, leaving the system in its previous state.

- **Consistency:** This ensures that a transaction brings the database from one valid state to another. It ensures that any data written to the database must be valid according to all defined rules, including constraints, cascades, triggers, and any combination thereof.

- **Isolation:** This ensures that the operations of a transaction are isolated from those of other transactions. This prevents transactions from interfering with each other and ensures that concurrent transactions yield consistent results.

- **Durability:** This ensures that once a transaction has been committed, it will remain so, even in the event of a system failure. This means that the changes made by the transaction are permanently recorded in the database.

Simulating a Transaction with SQL Statements

Let's simulate a transaction involving the borrowing of a book by a member from a library.

Setting Up the Environment:

First, let's create a BorrowedBooks table to track which books are borrowed by which members:

```
CREATE TABLE BorrowedBooks (
    borrow_id INT AUTO_INCREMENT PRIMARY KEY,
    book_id INT,
    member_id INT,
    borrow_date DATE,
    return_date DATE,
    FOREIGN KEY (book_id) REFERENCES Books(book_id),
    FOREIGN KEY (member_id) REFERENCES Members(member_id)
);
```

Simulating a Transaction with Explicit Locking

```
START TRANSACTION;
-- Step 1: Check if the book is available (not borrowed)
SELECT * FROM BorrowedBooks WHERE book_id = 1 AND return_date IS NULL FOR UPDATE;
-- Step 2: Borrow the book
INSERT INTO BorrowedBooks (book_id, member_id, borrow_date, return_date)
VALUES (1, 1, '2024-05-20', NULL);
```

-- Commit the transaction

COMMIT;

“FOR UPDATE” clause is used to lock the row being checked, ensuring no other transaction can modify it until the current transaction is completed.

Demonstrating Different Isolation Levels

Isolation levels control the visibility of data changes made by one transaction to other concurrent transactions. The standard isolation levels are:

- **READ UNCOMMITTED:** Lowest level, allowing dirty reads. One transaction may read data changed by another transaction that has not yet committed.
- **READ COMMITTED:** Prevents dirty reads. One transaction cannot read data from another transaction that has not yet committed.
- **REPEATABLE READ:** Prevents dirty reads and non-repeatable reads. Ensures that if a transaction reads the same row twice, it will see the same data both times, even if other transactions are modifying the data.
- **SERIALIZABLE:** Highest level, preventing dirty reads, non-repeatable reads, and phantom reads. Transactions are executed in a way that they appear to be serialized.