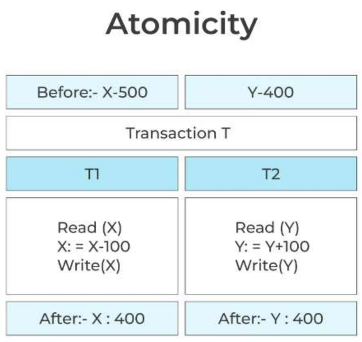
Name: V. Charan Kumar

Assignment 1: 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.

ACID stands for Atomicity, Consistency, Isolation, and Durability. These properties ensure that database transactions are reliable and maintain data integrity.

Atomicity: Atomicity ensures that a transaction is treated as a single unit of operation. It means that either all the operations within the transaction are successfully completed and the changes are committed, or none of the operations are performed and the database remains unchanged.



Consistency: Consistency ensures that the database remains in a consistent state before and after the transaction. It means that the database transitions from one consistent state to another consistent state after the transaction is executed.

Example:

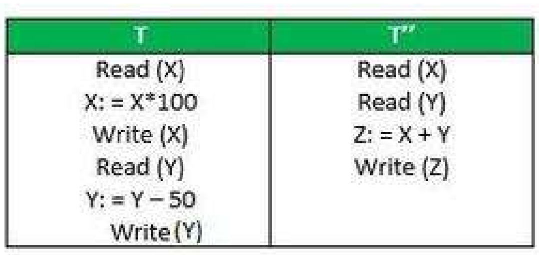
The total amount before and after the transaction must be maintained.

Total before T occurs = 500 + 200 = 700.

Total after T occurs = 400 + 300 = 700.

Therefore, the database is consistent. Inconsistency occurs in case T1 completes but T2 fails. As a result, T is incomplete.

Isolation: Isolation ensures that the execution of transactions concurrently does not result in data inconsistency. It means that each transaction appears to be executed in isolation from other transactions, even if they are executed concurrently.



T	T''
Read (X)	Read (X)
X: = X*100	Read (Y)
Write (X)	Z: = X + Y
Read (Y)	Write (Z)
Y: = Y – 50	
Write (Y)	

Durability: Durability ensures that once a transaction is committed, its changes are permanently saved and cannot be undone, even in the event of system failures.

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.

Query:

```
CREATE DATABASE library;

USE library;

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

CREATE TABLE books (
   book_id INT AUTO_INCREMENT PRIMARY KEY,
   title VARCHAR(200) NOT NULL,
   author_id INT,
   publication_year INT,
   FOREIGN KEY (author_id) REFERENCES authors(author_id)
);

CREATE TABLE Membership (
   member_id INT PRIMARY KEY,
   member_name VARCHAR(255) NOT NULL,
   email VARCHAR(255) UNIQUE NOT NULL,
   phone_number VARCHAR(20) NOT NULL
);

CREATE TABLE book_lends (
   l_id INT PRIMARY KEY,
   book_id INT,
   member_id INT,
   taken_date DATE,
   return_date DATE,
   returned BOOLEAN DEFAULT false,
   FOREIGN KEY (book_id) REFERENCES Books(book_id),
   FOREIGN KEY (member_id) REFERENCES Members(member_id));
```

Assignment 3: Compose SQL statements to BEGIN a transaction, INSERT a new record into the 'orders' table, COMMIT the transaction, then UPDATE the 'products' table, and ROLLBACK the transaction.

Queries:

```sql
BEGIN TRANSACTION;

INSERT INTO orders (order_id, customer_id, product_id, quantity, order_date)
VALUES (12345, 67890, 54321, 3, '2024-05-13');

COMMIT;

BEGIN TRANSACTION;

UPDATE products
SET quantity_in_stock = quantity_in_stock - 3
WHERE product_id = 54321;

ROLLBACK;
```

Assignment 4: Begin a transaction, perform a series of INSERTs into 'orders', setting a SAVEPOINT after each, rollback to the second SAVEPOINT, and COMMIT the overall transaction.

Queries:

```sql
BEGIN TRANSACTION;

INSERT INTO orders (order_id, product_id, quantity, order_date)
VALUES (1, 101, 3, '2024-05-13');

SAVEPOINT savepoint1;

INSERT INTO orders (order_id, product_id, quantity, order_date)
VALUES (2, 102, 2, '2024-05-14');

SAVEPOINT savepoint2;

INSERT INTO orders (order_id, product_id, quantity, order_date)
VALUES (3, 103, 1, '2024-05-15');

ROLLBACK TO SAVEPOINT savepoint2;

COMMIT;
```