

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.

ACID Properties of a Transaction

ACID properties ensure reliability and integrity in a database transaction:

1. **Atomicity:** A transaction is either fully completed or fully rolled back if any part of it fails.

Example: Transferring money between two accounts should either update both balances or none.

2. **Consistency:** The database should remain valid before and after the transaction, following defined rules and constraints.

Example: If money is withdrawn from one account, it must be credited to another, maintaining balance constraints.

3. **Isolation:** Transactions run independently, preventing one transaction from interfering with another. Different isolation levels help manage concurrency.

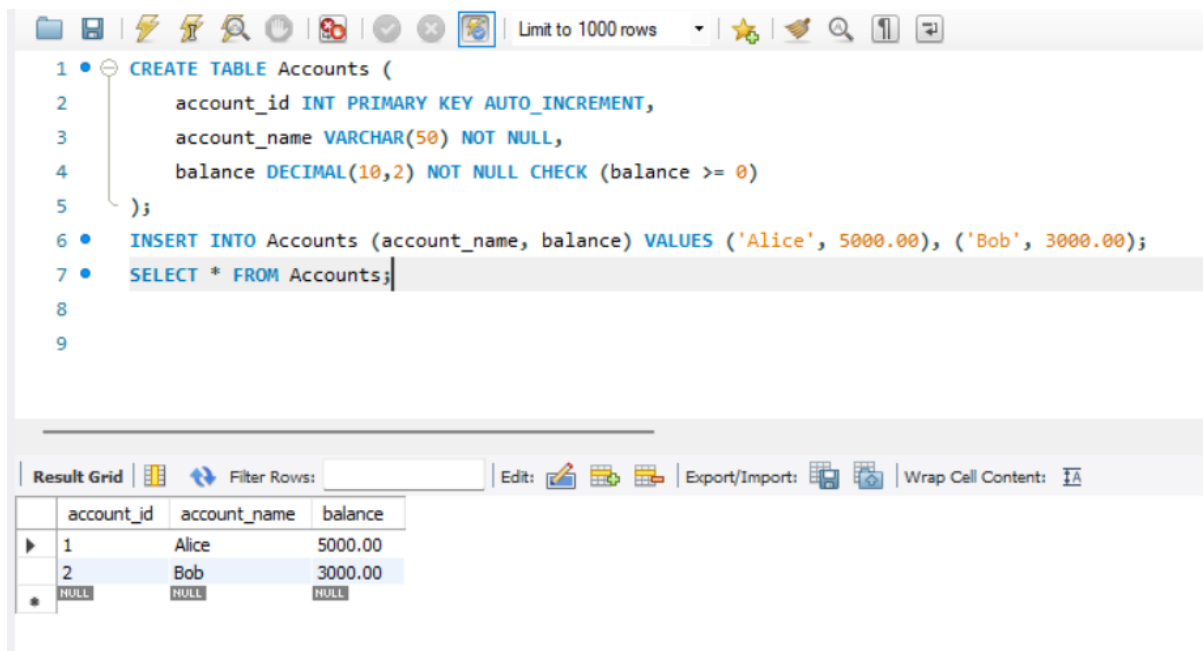
Example: If two users try to withdraw money from the same account, isolation ensures correct processing.

4. **Durability:** Once a transaction is committed, changes are permanently saved, even in case of system failure.

Example: A successful money transfer is not lost even if the database crashes immediately after.

Simulating a Transaction with Locking.

Step 1: Create Tables for a Banking System & Insert Sample Data.



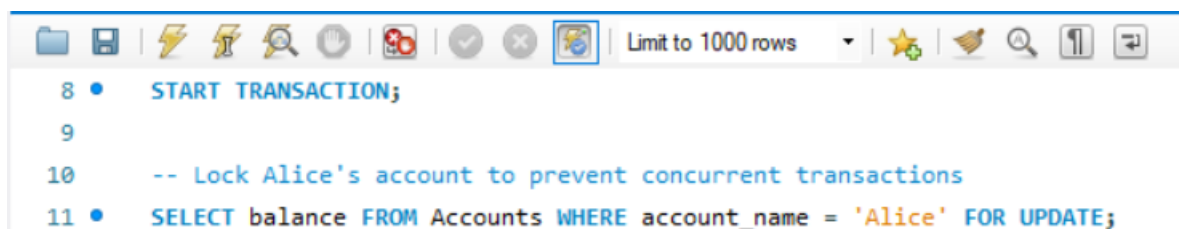
```

1 • CREATE TABLE Accounts (
2     account_id INT PRIMARY KEY AUTO_INCREMENT,
3     account_name VARCHAR(50) NOT NULL,
4     balance DECIMAL(10,2) NOT NULL CHECK (balance >= 0)
5 );
6 • INSERT INTO Accounts (account_name, balance) VALUES ('Alice', 5000.00), ('Bob', 3000.00);
7 • SELECT * FROM Accounts;
8
9

```

account_id	account_name	balance
1	Alice	5000.00
2	Bob	3000.00
* NULL	NULL	NULL

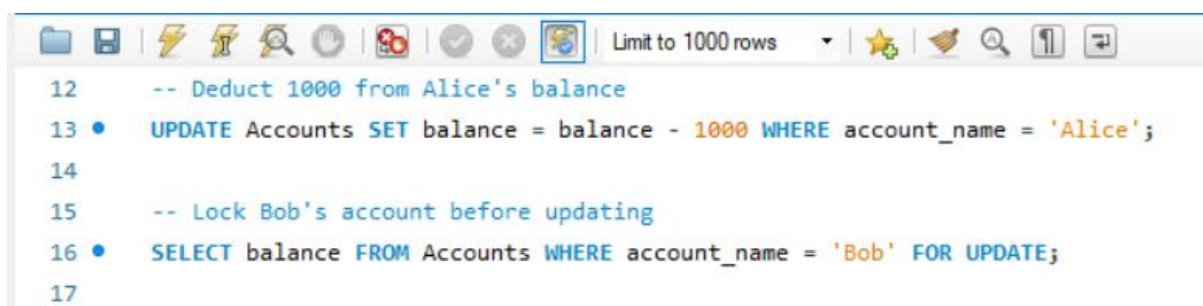
Step 2 : Perform a Transaction with Locking.



```

8 • START TRANSACTION;
9
10 -- Lock Alice's account to prevent concurrent transactions
11 • SELECT balance FROM Accounts WHERE account_name = 'Alice' FOR UPDATE;

```



```

12 -- Deduct 1000 from Alice's balance
13 • UPDATE Accounts SET balance = balance - 1000 WHERE account_name = 'Alice';
14
15 -- Lock Bob's account before updating
16 • SELECT balance FROM Accounts WHERE account_name = 'Bob' FOR UPDATE;
17

```



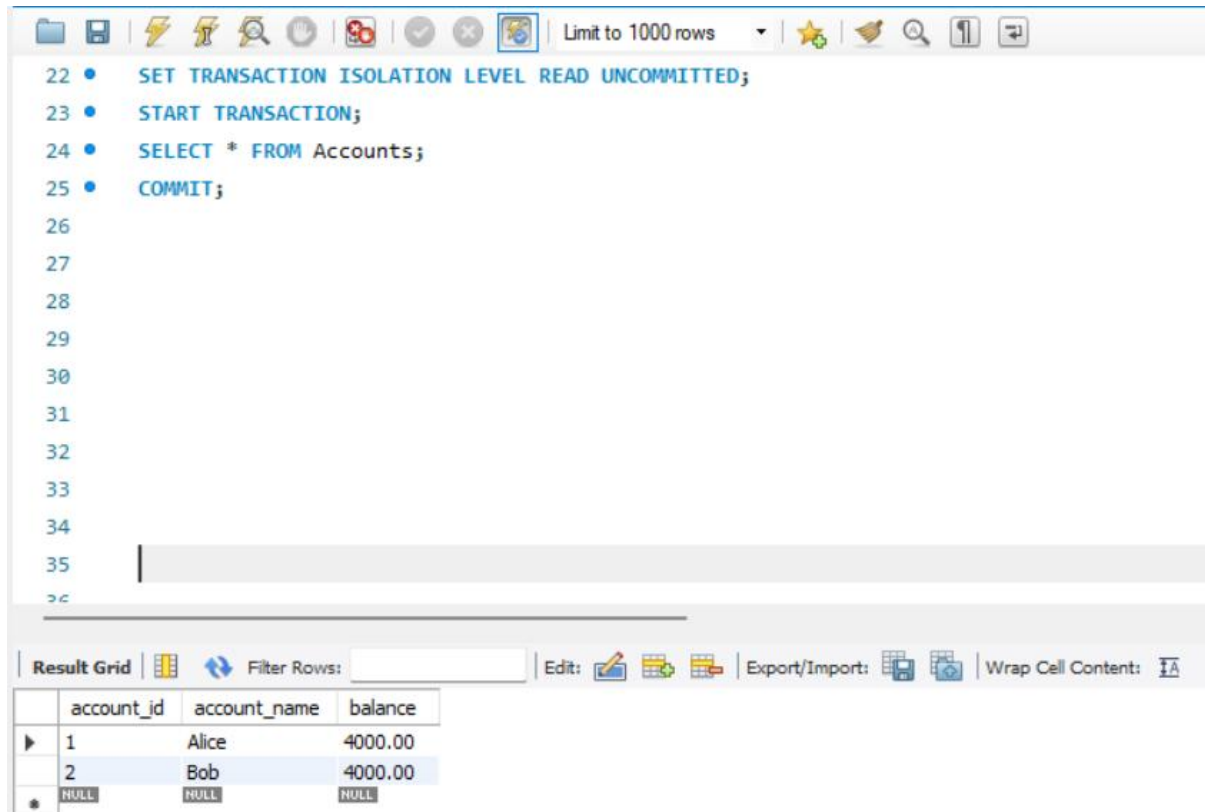
```

18 • UPDATE Accounts SET balance = balance + 1000 WHERE account_name = 'Bob';
19
20 -- Commit the transaction if everything is successful
21 • COMMIT;
22

```

Demonstrating Different Isolation Levels

Step 3 : Read Uncommitted (Lowest Isolation).



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 1000 rows' dropdown. The query window contains the following SQL code:

```
22 • SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;  
23 • START TRANSACTION;  
24 • SELECT * FROM Accounts;  
25 • COMMIT;  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35
```

Below the query window is the 'Result Grid' section. It includes a 'Filter Rows' input field and buttons for 'Edit', 'Export/Import', and 'Wrap Cell Content'. The result grid displays the following data:

	account_id	account_name	balance
▶	1	Alice	4000.00
	2	Bob	4000.00
*	NULL	NULL	NULL

Step 4 : Read Committed (Default in Many Databases).

The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a dropdown menu set to "Limit to 1000 rows". The query window contains the following SQL statements:

```
26 • SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
27 • START TRANSACTION;
28 • SELECT * FROM Accounts;
29 • COMMIT;
```

Below the query window is a "Result Grid" section. It includes a "Filter Rows:" input field and buttons for "Edit:", "Export/Import:", and "Wrap Cell Content:". The result grid displays the following data:

	account_id	account_name	balance
▶	1	Alice	4000.00
	2	Bob	4000.00
*	NULL	NULL	NULL

Step 5 : Repeatable Read.

The screenshot shows a SQL IDE interface. The query editor contains the following SQL code:

```
30 • SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
31 • START TRANSACTION;
32 • SELECT * FROM Accounts WHERE account_name = 'Alice';
33 • COMMIT;
```

The result grid below the query shows the following data:

	account_id	account_name	balance
▶	1	Alice	4000.00
*	NULL	NULL	NULL

Step 6 : Serializable (Highest Isolation).

The screenshot shows a SQL IDE interface. The query editor contains the following SQL code:

```
34 • SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
35 • START TRANSACTION;
36 • SELECT * FROM Accounts;
37 • COMMIT;
```

The result grid below the query shows the following data:

	account_id	account_name	balance
▶	1	Alice	4000.00
	2	Bob	4000.00
*	NULL	NULL	NULL