

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.

SOLUTION :

ACID PROPERTIES:

1. **Atomicity**: Atomicity ensures that either all operations within a transaction are successfully completed, or none of them are. It's like an "all-or-nothing" principle. If any part of the transaction fails, the entire transaction is rolled back, leaving the database in its original state.
2. **Consistency**: Consistency ensures that the database remains in a consistent state before and after the transaction. In other words, transactions should bring the database from one consistent state to another consistent state, preserving integrity constraints, foreign key relationships, etc.
3. **Isolation**: Isolation ensures that the execution of transactions concurrently does not result in any interference or inconsistency. Each transaction should appear to be executing in isolation, regardless of other concurrent transactions. Different isolation levels control the degree of isolation and trade-offs between concurrency and consistency.
4. **Durability**: Durability ensures that once a transaction is committed, its changes are permanent and will not be lost, even in the event of a system failure. This property ensures that the changes made by committed transactions persist even after a crash or restart.

```
CREATE TABLE BankAccount (  
    AccountID INT PRIMARY KEY,  
    Balance DECIMAL(10,2)  
);
```

```
INSERT INTO BankAccount (AccountID, Balance)
VALUES (1, 5000), (2, 6000);

UPDATE BankAccount SET Balance = Balance - 1000
WHERE AccountID = 1;

UPDATE BankAccount SET Balance = Balance + 1000
WHERE AccountID = 2;
```

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

SELECT * FROM BankAccount WHERE AccountID = 1;

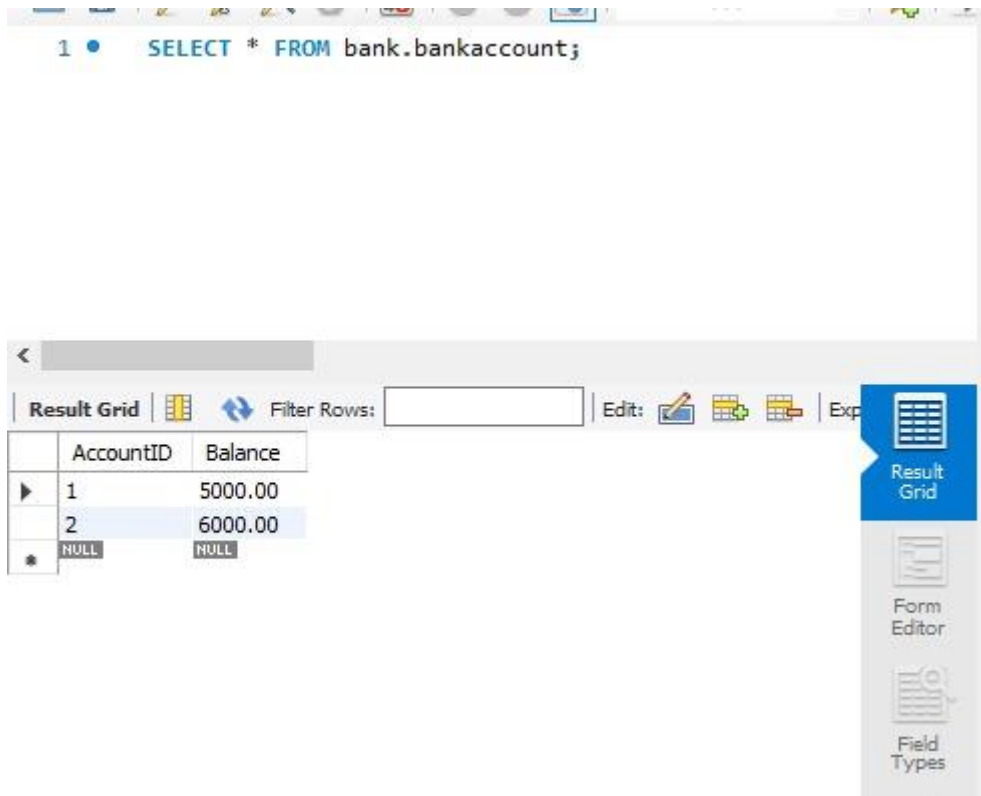
COMMIT;
```

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

```
UPDATE BankAccount SET Balance = Balance + 100 WHERE
AccountID = 1;

COMMIT;
```

OUTPUT:






The screenshot shows a database query tool interface. At the top, a SQL query is entered: `1 • SELECT * FROM bank.bankaccount;`. Below the query editor, there is a toolbar with icons for 'Result Grid', 'Filter Rows', 'Edit', and 'Export'. The 'Result Grid' icon is highlighted in blue. Below the toolbar, a table displays the query results. The table has two columns: 'AccountID' and 'Balance'. The first row shows '1' and '5000.00'. The second row shows '2' and '6000.00'. The third row shows 'NULL' and 'NULL'. To the right of the table, there is a vertical sidebar with three icons: 'Result Grid' (highlighted in blue), 'Form Editor', and 'Field Types'.

	AccountID	Balance
▶	1	5000.00
	2	6000.00
*	NULL	NULL

1 • `SELECT * FROM bank.bankaccount;`

Result Grid

Filter Rows:

Edit:    Exp

	AccountID	Balance
▶	1	4100.00
	2	7000.00
•	NULL	NULL

Result Grid

Form Editor

Field Types




Query Stats

Limit to 1000 rows

1 • `SELECT * FROM bank.bankaccount;`

Result Grid

Filter Rows:

Edit:    Exp

	AccountID	Balance
▶	1	4000.00
	2	6000.00
•	NULL	NULL

Result Grid

Form Editor

Field Types