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.

The **ACID** properties are essential characteristics that ensure reliable processing of database transactions.

1. **Atomicity :-** Ensures that a transaction is all-or-nothing. Either all operations within the transaction are completed successfully, or none are. If any part of the transaction fails, the entire transaction is rolled back to its previous state.

2. **Consistency :-** Ensures that a transaction brings the database from one valid state to another. The database should always comply with its defined rules, constraints, and validations, both before and after the transaction.

3. **Isolation :-** Ensures that transactions are securely and independently processed. Transactions occurring concurrently do not affect each other's execution and outcomes. This means the intermediate states of a transaction are invisible to other transactions.

4. **Durability :-** Ensures that once a transaction has been committed, it remains so, even in the event of a system failure. The changes made by the transaction are permanently saved in the database.

Queries :-

```
create database Assign;

use Assign;

create table accounts (

accountid int primary key,

balance decimal(10, 2) not null

);

insert into accounts (accountid, balance) values (1, 2000.00), (2, 3000.00);

create table transactions (

transactionid int primary key auto_increment,

accountid int,

amount decimal(10, 2),

transactiondate timestamp default current_timestamp,

FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)

);

SELECT * FROM assign.accounts;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

accountid	balance
1	2000.00
2	3000.00
NULL	NULL

START TRANSACTION;

-- Lock the rows for both accounts to prevent concurrent updates

select balance from accounts where accountid = 1 for update;

select balance from accounts where accountid = 2 for update;

-- Perform the transfer

update accounts set balance = balance - 200.00 where accountid = 1;

update accounts set balance = balance + 500.00 where accountid = 2;

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

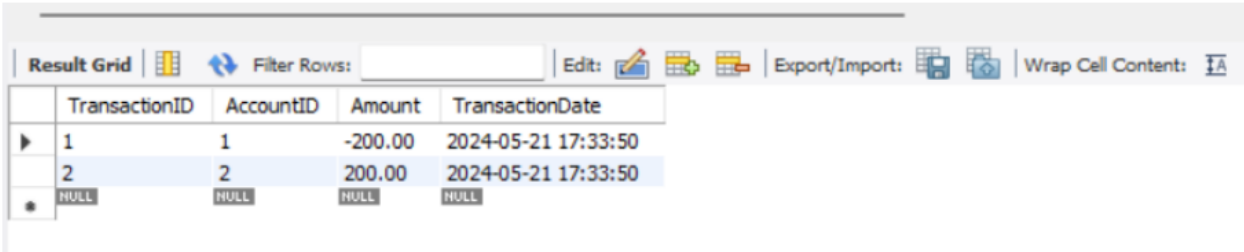
AccountID	Balance
1	1800.00
2	3500.00
NULL	NULL

-- Log the transaction

insert into transactions (accountid, amount) values (1, -200.00), (2, 200.00);

-- Commit the transaction

COMMIT;



TransactionID	AccountID	Amount	TransactionDate
1	1	-200.00	2024-05-21 17:33:50
2	2	200.00	2024-05-21 17:33:50
NULL	NULL	NULL	NULL

Demonstrating Different Isolation Levels:

1. Read Uncommitted

-- Transaction 1

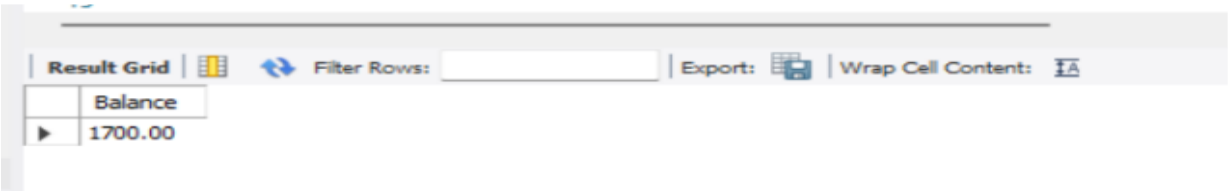
UPDATE Accounts SET Balance = Balance - 100.00 WHERE AccountID = 1;

-- Transaction 2 (dirty read possible)

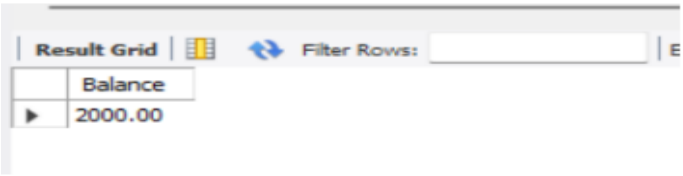
SELECT Balance FROM Accounts WHERE AccountID = 1;

-- Transaction 1

ROLLBACK;



Balance
1700.00



Balance
2000.00

2. Read Committed

--- Transaction 1

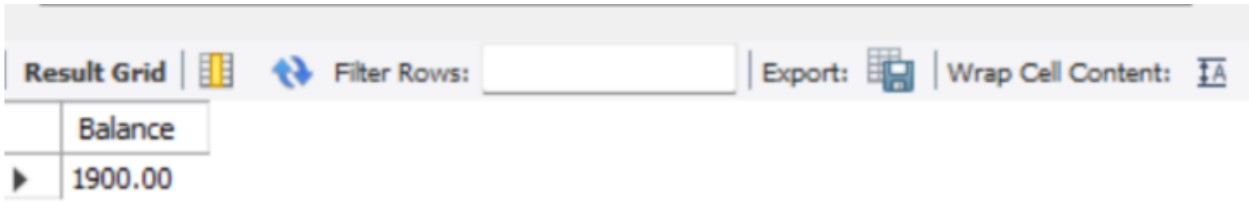
UPDATE Accounts SET Balance = Balance - 100.00 WHERE AccountID = 1;

-- Transaction 2 (will not see the uncommitted change)

SELECT Balance FROM Accounts WHERE AccountID = 1;

-- Transaction 1

COMMIT;



3. Repeatable Read

-- Transaction 1

SELECT Balance FROM Accounts WHERE AccountID = 1;

-- Transaction 2

UPDATE Accounts SET Balance = Balance - 100.00 WHERE AccountID = 1;

COMMIT;

-- Transaction 1 (will see the original balance)

SELECT Balance FROM Accounts WHERE AccountID = 1;

COMMIT;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	Balance
▶	1800.00

4. Serializable

-- Transaction 1

SELECT Balance FROM Accounts WHERE AccountID = 1;

-- Transaction 2 (will be blocked until Transaction 1 is completed

UPDATE Accounts SET Balance = Balance - 100.00 WHERE AccountID = 1;

COMMIT;

-- Transaction 1

COMMIT;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	Balance
▶	1600.00