Advanced Database Topics (COMP-8157)

Lab-4

Name: Karan Mahajan
Section: 3
Student ID: 110119774

Implement a database application that simulates the concurrent access to a shared database table by multiple clients. The database table should store information about a bank account, including account number, account holder name, and balance. The application should allow clients to perform the following operations:

1. Deposit money into the account
2. Withdraw money from the account
3. Check the balance of the account

```sql
CREATE TABLE KaranMahajanBankAccounts (
AccNumber INT PRIMARY KEY,
AccHolder NVARCHAR(255),
Balance DECIMAL(10, 2)
);
```

Messages

4:27:38 PM Started executing query at Line 1
Commands completed successfully.
Total execution time: 00:00:00.022

```sql
INSERT INTO KaranMahajanBankAccounts (AccNumber, AccHolder, Balance)
VALUES
(4, 'Karan Mahajan', 11500.00),
(3, 'Ansh Aggarwal', 12000.00),
(2, 'Niharika', 18000.00),
(1, 'Yash Batliwala', 40000.00);
```

ssages

4:29:02 PM Started executing query at Line 7
(4 rows affected)
Total execution time: 00:00:00.021

Procedures to deposit, withdraw and view the account balance

```
-- Procedure to deposit money
CREATE OR ALTER PROCEDURE KaranMahajanDepositMoney
    @AccountNumber INT,
    @Money DECIMAL(10, 2)
AS
BEGIN
    UPDATE KaranMahajanBankAccounts
    SET Balance = Balance + @Money
    WHERE AccNumber = @AccountNumber
END

-- Procedure to withdraw money
CREATE OR ALTER PROCEDURE KaranMahajanWithdrawMoney
    @AccountNumber INT,
    @Money DECIMAL(10, 2)
AS
BEGIN
    UPDATE KaranMahajanBankAccounts
    SET Balance = Balance - @Money
    WHERE AccNumber = @AccountNumber AND Balance >= @Money
END


-- Procedure to check the account balance
CREATE OR ALTER PROCEDURE KaranMahajanViewBalance
    @AccountNumber INT
AS
BEGIN
    SELECT balance from KaranMahajanBankAccounts
    WHERE AccNumber = @AccountNumber;
END
```

```
53    END
54
55    BEGIN TRANSACTION
56    Waitfor Delay '00:00:07'
57    EXEC KaranMahajanDepositMoney 1, 1000.00
58    SELECT * from KaranMahajanBankAccounts;
59    EXEC KaranMahajanDepositMoney 1, 100.00
60    SELECT * from KaranMahajanBankAccounts;
61    EXEC KaranMahajanWithdrawMoney 1, 200.00
62    SELECT * from KaranMahajanBankAccounts;
63    EXEC KaranMahajanViewBalance 1
64    COMMIT;
```

Results Messages

	AccNumber	AccHolder	Balance
1	1	Yash Batliwala	1000.00
2	2	Niharika	18000.00
3	3	Ansh Aggarwal	12000.00
4	4	Karan Mahajan	11500.00

	AccNumber	AccHolder	Balance
1	1	Yash Batliwala	1100.00
2	2	Niharika	18000.00
3	3	Ansh Aggarwal	12000.00
4	4	Karan Mahajan	11500.00

	AccNumber	AccHolder	Balance
1	1	Yash Batliwala	900.00
2	2	Niharika	18000.00
3	3	Ansh Aggarwal	12000.00
4	4	Karan Mahajan	11500.00

	balance
1	900.00

The result after the 1st transaction

```
1   BEGIN TRANSACTION
2   WAITFOR DELAY '00:00:02'
3   EXEC KaranMahajanWithdrawMoney 1,1000
4   SELECT * from KaranMahajanBankAccounts;
5   COMMIT;
6
```

Results | Messages

	AccNumber	AccHolder	Balance
1	1	Yash Batliwala	0.00
2	2	Niharika	18000.00
3	3	Ansh Aggarwal	12000.00
4	4	Karan Mahajan	11500.00

The result after the 2nd transaction

```
6
7   SELECT * from KaranMahajanBankAccounts;
```

Results | Messages

	AccNumber	AccHolder	Balance
1	1	Yash Batliwala	900.00
2	2	Niharika	18000.00
3	3	Ansh Aggarwal	12000.00
4	4	Karan Mahajan	11500.00

The actual value for the balance

We can see that we are running two Transactions concurrently in the screenshots up top. If Transaction 1 produces the right output, everything is well. However, transaction 2 is not giving us the right output since, according to transaction 1, there would not be enough amount to withdraw from AccountNumber = 1. Instead, it is reading the previous amount of AccountNumber = 1, which is incorrect. This is a database concurrency issue.

We will now use lock-based synchronization, a well-known concurrency control approach, to resolve the problem. We're going to revise our process and execute both transactions simultaneously once more.

```
-- Updated Procedure to deposit money
CREATE OR ALTER PROCEDURE KaranMahajanDepositMoney
    @AccountNumber INT,
    @Money DECIMAL(10, 2)
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION

        SELECT *
        FROM KaranMahajanBankAccounts with (UPDLOCK, HOLDLOCK)
        WHERE AccNumber = @AccountNumber;

        -- Update the balance
        UPDATE KaranMahajanBankAccounts
        SET Balance = Balance + @Money
        WHERE AccNumber = @AccountNumber;

        COMMIT;
    END TRY
    BEGIN CATCH
        ROLLBACK TRANSACTION;
    END CATCH
```

```
END;



-- Updated Procedure to withdraw money
CREATE OR ALTER PROCEDURE KaranMahajanDepositMoney
    @AccountNumber INT,
    @Money DECIMAL(10, 2)
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION;

        DECLARE @currentBalance DECIMAL(10,2)
        SELECT @currentBalance = Balance
        FROM KaranMahajanBankAccounts with (UPDLOCK, HOLDLOCK)
        WHERE AccNumber = @AccountNumber;

        IF @Money > @currentBalance
        BEGIN
            THROW 50000, 'INSUFFICIENT BALANCE', 1;
            ROLLBACK TRANSACTION;
            RETURN;
        END

        -- Update the balance
        UPDATE KaranMahajanBankAccounts
        SET Balance = Balance - @Money
        WHERE AccNumber = @AccountNumber;

        COMMIT;
    END TRY
    BEGIN CATCH
        ROLLBACK TRANSACTION;
    END CATCH
END;



-- Updated Procedure to VIEW money
CREATE OR ALTER PROCEDURE KaranMahajanDepositMoney
    @AccountNumber INT,
```

```
    @Money DECIMAL(10, 2)
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION;

        SELECT *
        FROM KaranMahajanBankAccounts with (UPDLOCK, HOLDLOCK)
        WHERE AccNumber = @AccountNumber;


        DECLARE @currentBalance DECIMAL(10,2)
        SELECT @currentBalance = Balance
        FROM KaranMahajanBankAccounts
        WHERE AccNumber = @AccountNumber;

        COMMIT;
    END TRY
    BEGIN CATCH
        ROLLBACK TRANSACTION;
    END CATCH
END;
```

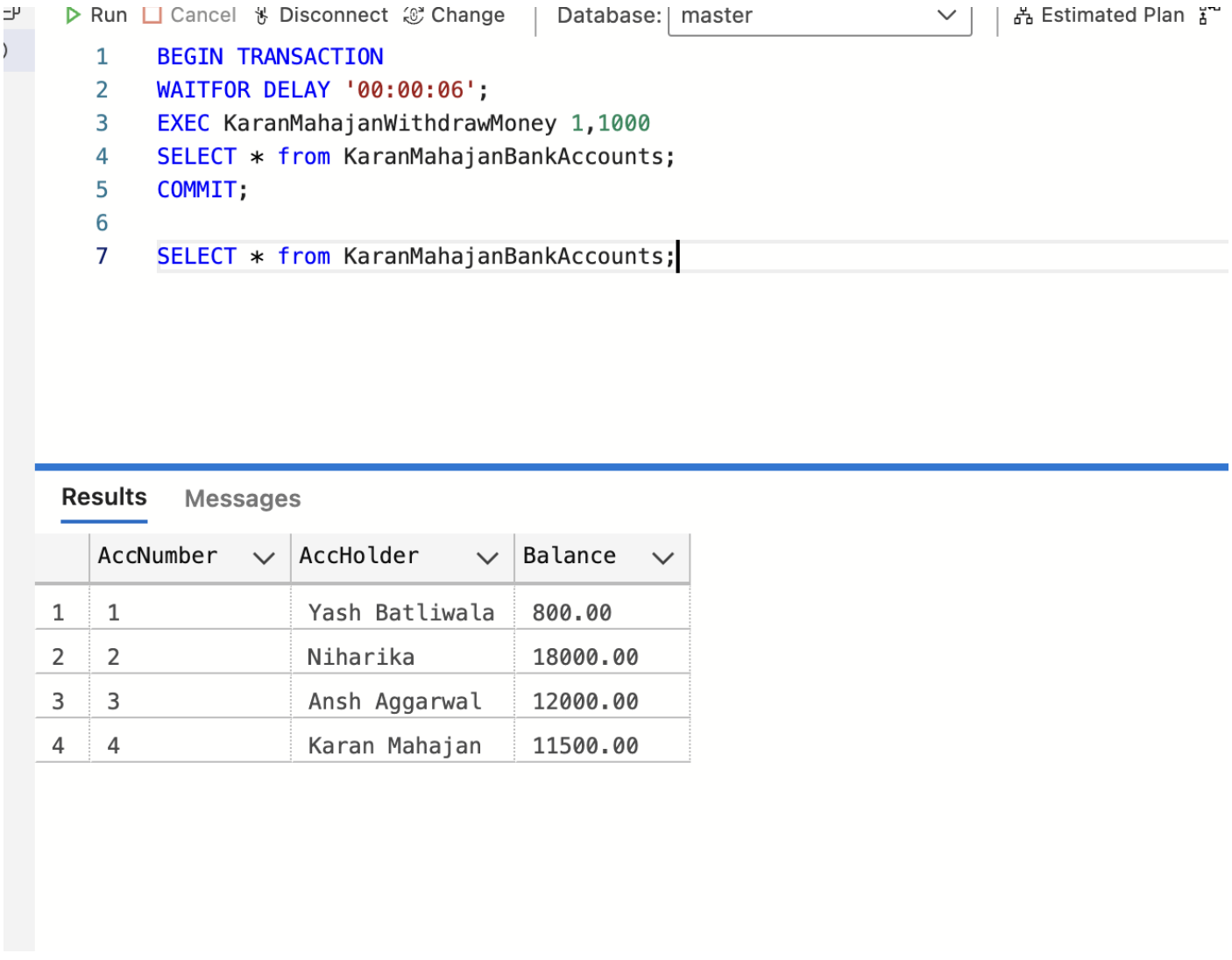
```
161
162    BEGIN TRANSACTION
163    WAITFOR DELAY '00:00:05';
164    EXEC KaranMahajanDepositMoney 1, 1000.00;
165    SELECT * from KaranMahajanBankAccounts;
166    EXEC KaranMahajanDepositMoney 1, 100.00;
167    SELECT * from KaranMahajanBankAccounts;
168    EXEC KaranMahajanWithdrawMoney 1, 200.00;
169    SELECT * from KaranMahajanBankAccounts;
170    EXEC KaranMahajanViewBalance 1;
171    COMMIT;
172
173
174    SELECT * from KaranMahajanBankAccounts;
```

Results Messages

AccNumber	AccHolder	Balance
1	Yash Batliwala	1000.00

AccNumber	AccHolder	Balance
1	Yash Batliwala	1000.00
2	Niharika	18000.00
3	Ansh Aggarwal	12000.00
4	Karan Mahajan	11500.00

AccNumber	AccHolder	Balance



Explanation: The above screenshot shows that the concurrency between the transactions is handled perfectly using the implementation of the lock-based synchronization.