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

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

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

A screenshot of a computer

Description automatically generated

The result after the 1st transaction

A screenshot of a computer

Description automatically generated

The result after the 2nd transaction

A screenshot of a computer

Description automatically generated

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;

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

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