University of Windsor

School of Computer Science

Masters in Applied Computing (M.A.C)

Subject Code: COMP8157

Subject Name: Advanced Database Topics

Professor Dr Shafaq Khan

Lab 4

by

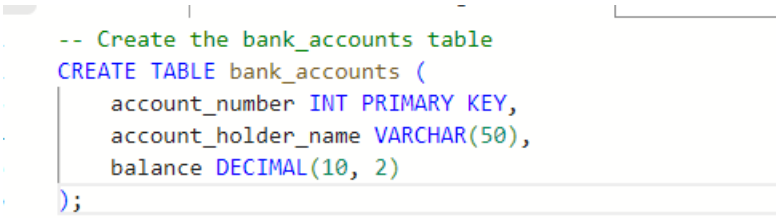
Hamza Baig (110089314)

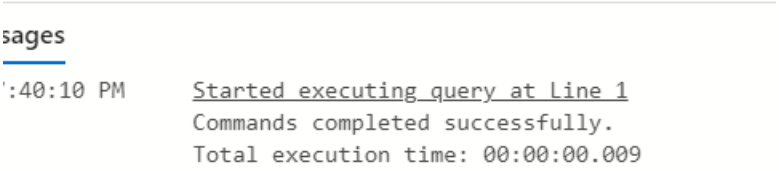
Table of Contents

Database:

bank_account database:

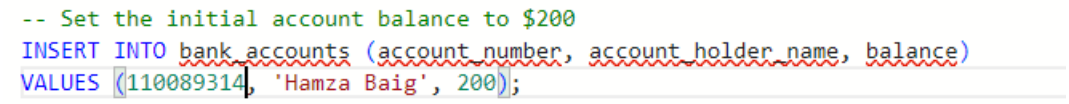
```
-- Create the bank_accounts table
CREATE TABLE bank_accounts (
    account_number INT PRIMARY KEY,
    account_holder_name VARCHAR(50),
    balance DECIMAL(10, 2)
);
```

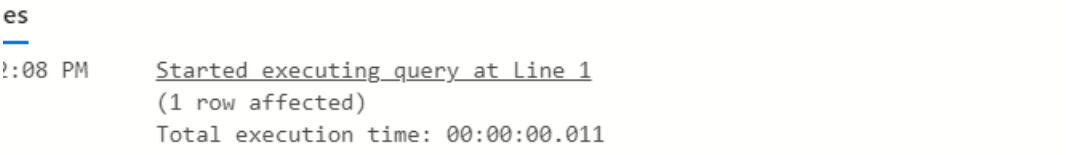




Insert initial account information:

```
-- Set the initial account balance to $200
INSERT INTO bank_accounts (account_number, account_holder_name, balance)
VALUES (110089314, 'Hamza Baig', 200);
```






```
-- Show bank accounts
SELECT * FROM bank_accounts;
```

```
11
12    SELECT * FROM bank_accounts;
```

Results | Messages

	account_number	account_holder_name	balance
1	110089314	Hamza Baig	200.00

Methods for database:

I have used isolation level repeatable read, as it would block the second transaction and let the first one finish first, in this way there will be no conflict, or lost updates, as one transaction completes first.

Deposit:

Code:

```
-- Deposit money
DECLARE @deposit AS INT=200;

DECLARE @balance AS DECIMAL(10, 2);
SET @balance=isnull((SELECT balance FROM bank_accounts WHERE account_number = 110089314),0);
-- Transaction 2
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
UPDATE bank_accounts SET balance = @balance+@deposit WHERE account_number = 110089314;
```

Run | Cancel | Disconnect | Change Connection | master | Estimated Plan | Enable A

```sql
-- Deposit money
DECLARE @deposit AS INT=200;

DECLARE @balance AS DECIMAL(10, 2);
SET @balance=isnull((SELECT balance FROM bank_accounts WHERE account_number = 110089314),0);
-- Transaction 2
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
UPDATE bank_accounts SET balance = @balance+@deposit WHERE account_number = 110089314;

SELECT * FROM bank_accounts;

-- Withdraw money
DECLARE @withdraw AS INT=800;

DECLARE @current_balance AS DECIMAL(10, 2);
SET @current_balance=isnull((SELECT balance FROM bank_accounts WHERE account_number = 110089314)
-- Transaction 2
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
IF (@current_balance-@withdraw>0)
    UPDATE bank_accounts SET balance = @current_balance-@withdraw WHERE account_number = 110089
ELSE
    PRINT 'You do not have enough money in your account!';
```

Results | Messages

	account_number	account_holder_name	balance
	110089314	Hamza Baig	1100.00

```sql
-- Transaction 1
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
BEGIN TRANSACTION
SELECT balance FROM bank_accounts WHERE account_number = 110089314
-- Do Some work
WAITFOR DELAY '00:00:10'
SELECT balance FROM bank_accounts WHERE account_number = 110089314
COMMIT TRANSACTION
```

Results | Messages

	balance
1	900.00

From the above images, it is evident that the transaction 1 gives the old balance and meanwhile, transaction 2 is blocked, so it executes and gives the updated values.

Withdrawal:

Code:

```
-- Withdraw money
DECLARE @withdraw AS INT=800;

DECLARE @current_balance AS DECIMAL(10, 2);
SET @current_balance=isnull((SELECT balance FROM bank_accounts WHERE account_number =
110089314),0);

-- Transaction 2
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
IF (@current_balance-@withdraw>0)
    UPDATE bank_accounts SET balance = @current_balance-@withdraw WHERE account_number =
110089314;
ELSE
    PRINT 'You do not have enough money in your account!';
```

```
15    -- Transaction 1
16    SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
17    BEGIN TRANSACTION
18    SELECT balance FROM bank_accounts WHERE account_number = 110089314
19    -- Do Some work
20    WAITFOR DELAY '00:00:10'
21    SELECT balance FROM bank_accounts WHERE account_number = 110089314
22    COMMIT TRANSACTION
```

Results Messages

	balance
1	1100.00

```
24  -- Display balance
25  SELECT balance FROM bank_accounts WHERE account_number = 110089314;
26
```

Results Messages

	balance
1	300.00

The above example also demonstrates how isolation level repeatable read is blocking the second transaction and completing the first transaction first.

In case of error (withdrawing money you don't have):

```
DECLARE @withdraw AS INT=800;

DECLARE @current_balance AS DECIMAL(10, 2);
SET @current_balance=isnull((SELECT balance FROM bank_accounts WHERE account_number = 110089314),0);
-- Transaction 2
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
IF (@current_balance-@withdraw>0)
    UPDATE bank_accounts SET balance = @current_balance-@withdraw WHERE account_number = 110089314;
ELSE
    PRINT 'You do not have enough money in your account!';

-- Display balance
```

```
es
4:18 PM     Started executing query at Line 13
            You do not have enough money in your account!
            Total execution time: 00:00:00.014
```

View Statement:

Code:

```
-- Display balance
SELECT balance FROM bank_accounts WHERE account_number = 110089314;
```

```
-- Transaction 1
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
BEGIN TRANSACTION
SELECT balance FROM bank_accounts WHERE account_number = 110089314
-- Do Some work
WAITFOR DELAY '00:00:10'
SELECT balance FROM bank_accounts WHERE account_number = 110089314
COMMIT TRANSACTION
```

```
24  -- Display balance
25  SELECT balance FROM bank_accounts WHERE account_number = 110089314;
26
```

Results Messages

	balance
1	300.00