Development of Banking System Database

Project Description

The task involves designing, implementing and testing a database schema for a banking system. The database manages customer information, account and transaction data.

Introduction

This project aims to design and implement a robust database schema for a banking system, ensuring comprehensive management of customer information, accounts, transactions, and other relevant data. The developed database will serve as the foundation for banking operations, providing structured storage and easy retrieval of critical financial information while maintaining data integrity and security. By implementing a well-structured relational database, this project ensures efficient data handling, accurate transaction tracking, and enhanced customer service capabilities.

Database Schema

Tables and Definitions

1. Customers Table

Column Name	Data Type	Constraints
CustomerID	INT	PRIMARY KEY
FirstName	VARCHAR(50)	NOT NULL
LastName	VARCHAR(50)	NOT NULL
DateOfBirth	DATE	
Email	VARCHAR(100)	UNIQUE
Phone	VARCHAR(15)	UNIQUE
Address	VARCHAR(255)	

2. Accounts Table

Column Name	Data Type	Constraints
AccountNumber	INT	PRIMARY KEY
CustomerID	INT	FOREIGN KEY REFERENCES Customers(CustomerID)
AccountType	VARCHAR(50)	
Balance	DECIMAL(10, 2)	CHECK (Balance >= 0)

3. Transactions Table

Column Name	Data Type	Constraints
TransactionID	INT	PRIMARY KEY
AccountNumber	INT	FOREIGN KEY REFERENCES Accounts(AccountNumber)
TransactionType	VARCHAR(50)	
Amount	DECIMAL(10, 2)	CHECK (Amount > 0)
TransactionDate	DATETIME	

Constraints

1. Customers Table:
    a. Primary Key: CustomerID
    b. Unique: Email, Phone
    c. Not Null: FirstName, LastName
2. Accounts Table:
    a. Primary Key: AccountNumber
    b. Foreign Key: CustomerID references Customers(CustomerID)
    c. Check: Balance >= 0
3. Transactions Table:
    a. - Primary Key: TransactionID
    b. - Foreign Key: AccountNumber references Accounts(AccountNumber)
    c. - Check: Amount > 0

Stored Procedures

1. Creating Customers

```
CREATE PROCEDURE CreateCustomer
    @FirstName NVARCHAR(50),
    @LastName NVARCHAR(50),
    @DateOfBirth DATE,
    @Email NVARCHAR(100),
    @Phone NVARCHAR(15),
    @Address NVARCHAR(255)
AS
BEGIN
    INSERT INTO Customers (FirstName, LastName, DateOfBirth, Email, Phone, Address)
    VALUES (@FirstName, @LastName, @DateOfBirth, @Email, @Phone, @Address);
END;
```

2. Opening Accounts

```
CREATE PROCEDURE OpenAccount
    @CustomerID INT,
    @AccountType NVARCHAR(50),
    @InitialDeposit DECIMAL(18, 2)
AS
BEGIN
    INSERT INTO Accounts (CustomerID, AccountType, Balance)
    VALUES (@CustomerID, @AccountType, @InitialDeposit);
END;
```

3. Depositing Money

```
CREATE PROCEDURE DepositMoney
    @AccountNumber INT,
    @Amount DECIMAL(18, 2)
AS
BEGIN
    UPDATE Accounts
    SET Balance = Balance + @Amount
    WHERE AccountNumber = @AccountNumber;

    INSERT INTO Transactions (AccountNumber, TransactionType, Amount, TransactionDate)
    VALUES (@AccountNumber, 'Deposit', @Amount, GETDATE());
END;
```

4. Withdrawing Money

```
CREATE PROCEDURE WithdrawMoney
    @AccountNumber INT,
    @Amount DECIMAL(18, 2)
AS
BEGIN
    UPDATE Accounts
    SET Balance = Balance - @Amount
    WHERE AccountNumber = @AccountNumber;

    INSERT INTO Transactions (AccountNumber, TransactionType, Amount, TransactionDate)
    VALUES (@AccountNumber, 'Withdrawal', @Amount, GETDATE());
END;
```

5. Transferring Amount

```
CREATE PROCEDURE TransferMoney
    @FromAccount INT,
    @ToAccount INT,
    @Amount DECIMAL(18, 2)
AS
BEGIN
    UPDATE Accounts
    SET Balance = Balance - @Amount
    WHERE AccountNumber = @FromAccount;

    UPDATE Accounts
    SET Balance = Balance + @Amount
    WHERE AccountNumber = @ToAccount;

    INSERT INTO Transactions (AccountNumber, TransactionType, Amount, TransactionDate)
    VALUES (@FromAccount, 'Transfer Out', @Amount, GETDATE());

    INSERT INTO Transactions (AccountNumber, TransactionType, Amount, TransactionDate)
    VALUES (@ToAccount, 'Transfer In', @Amount, GETDATE());
END;
```

6. Viewing Transaction History

```
CREATE PROCEDURE ViewTransactionHistory
    @AccountNumber INT
AS
BEGIN
    SELECT TransactionID, TransactionType, Amount, TransactionDate
    FROM Transactions
    WHERE AccountNumber = @AccountNumber
    ORDER BY TransactionDate DESC;
END;
```

Testing Procedures

1. Creating a Customer:
    a. Ensure all fields are correctly inserted.
    b. Test for unique constraints on Email and Phone.

```
EXEC CreateCustomer @CustomerID=103, @FirstName = 'Mukesh', @LastName = 'Tyagi', @DateOfBirth = '1980-02-10',
@Email = 'mukesh.tyagi@example.com', @Phone = '1234567890', @Address = '123, Shanti Vihar';

EXEC CreateCustomer @CustomerID=108, @FirstName = 'Rishi', @LastName = 'Sharma', @DateOfBirth = '1990-08-09',
@Email = 'rishi.sharma@example.com', @Phone = '0987654321', @Address = '456, Aarya Road';
```

121 %

Messages

```
(1 row affected)

(1 row affected)

Completion time: 2024-07-29T11:24:51.5972569+05:30
```

```
-- Trying to insert duplicate Email (should fail)
EXEC CreateCustomer @CustomerID = '111', @FirstName = 'ABC', @LastName = 'abc', @DateOfBirth = '1995-10-01',
@Email = 'mukesh.tyagi@example.com', @Phone = '1111111111', @Address = '789 Vasant Vihar';
```

121 %

Messages

```
Msg 2627, Level 14, State 1, Procedure CreateCustomer, Line 11 [Batch Start Line 0]
Violation of UNIQUE KEY constraint 'UQ_Customers_Email'. Cannot insert duplicate key in object 'dbo.Customers'. The duplicate key value is (mukesh.tyagi@example.c
The statement has been terminated.

Completion time: 2024-07-29T11:35:44.0280868+05:30
```

```
-- Trying to insert duplicate Phone (should fail)
EXEC CreateCustomer @CustomerID = '111',@FirstName = 'DEF', @LastName = 'def', @DateOfBirth = '1995-10-01',
@Email = 'defdef@example.com', @Phone = '1234567890', @Address = '144 Shiv Colony';
```

121 %

Messages

```
Msg 2627, Level 14, State 1, Procedure CreateCustomer, Line 11 [Batch Start Line 0]
Violation of UNIQUE KEY constraint 'UQ_Customers_Phone'. Cannot insert duplicate key in object 'dbo.Customers'. The duplicate key value is (1234567890).
The statement has been terminated.

Completion time: 2024-07-29T11:39:17.6122437+05:30
```

2. Opening an Account:
    a. Ensure the account is linked to the correct customer.
    b. Check the initial deposit balance.

```
EXEC OpenAccount @AccountNumber = '100001', @CustomerID = '103', @AccountType = 'Savings',
@InitialDeposit = 1000.00;
EXEC OpenAccount @AccountNumber = '100004', @CustomerID = '108', @AccountType = 'Checking',
@InitialDeposit = 500.00;
```

133 %

Messages

```
(1 row affected)

(1 row affected)

Completion time: 2024-07-29T12:28:14.1271724+05:30
```

3. Depositing Money:
    a. Verify the balance is updated correctly.
    b. Check the transaction record.

```
EXEC DepositMoney @AccountNumber = 100001, @Amount = 200.00;
SELECT * FROM Accounts WHERE AccountNumber = 100001; -- Check the updated balance
SELECT * FROM Transactions WHERE AccountNumber = 100001; -- Check the transaction record
```

146 %

Results Messages

	AccountNumber	CustomerID	AccountType	Balance
1	100001	103	Savings	1200.00

	TransactionID	AccountNumber	TransactionType	Amount	TransactionDate
1	1	100001	Transfer Out	100.00	2024-07-29 14:37:18.890
2	4	100001	Deposit	200.00	2024-07-29 14:48:08.650
3	5	100001	Deposit	200.00	2024-07-29 14:48:26.123

4. Withdrawing Money:
    a. Ensure sufficient balance before withdrawal.
    b. Verify the balance is updated correctly.
    c. Check the transaction record.

```
EXEC WithdrawMoney @AccountNumber = 100001, @Amount = 150.00;
SELECT * FROM Accounts WHERE AccountNumber = 100001; -- Check the updated balance
SELECT * FROM Transactions WHERE AccountNumber = 100001; -- Check the transaction record
```

146 %

Results | Messages

	AccountNumber	CustomerID	AccountType	Balance
1	100001	103	Savings	1050.00

	TransactionID	AccountNumber	TransactionType	Amount	TransactionDate
1	1	100001	Transfer Out	100.00	2024-07-29 14:37:18.890
2	4	100001	Deposit	200.00	2024-07-29 14:48:08.650
3	5	100001	Deposit	200.00	2024-07-29 14:48:26.123
4	6	100001	Withdrawal	150.00	2024-07-29 14:51:52.143

5. Transferring Amount:
    a. Ensure sufficient balance in the from account.
    b. Verify both accounts are updated correctly.
    c. Check both transaction records.

```
EXEC TransferMoney @FromAccount = 100001, @ToAccount = 100004, @Amount = 100.00;
SELECT * FROM Accounts WHERE AccountNumber = 100001; -- Check the updated balance
SELECT * FROM Accounts WHERE AccountNumber = 100004; -- Check the updated balance
SELECT * FROM Transactions WHERE AccountNumber = 100001; -- Check the transaction record
SELECT * FROM Transactions WHERE AccountNumber = 100004; -- Check the transaction record
```

46 %

Results | Messages

	AccountNumber	CustomerID	AccountType	Balance
1	100001	103	Savings	800.00

	AccountNumber	CustomerID	AccountType	Balance
1	100004	108	Checking	700.00

	TransactionID	AccountNumber	TransactionType	Amount	TransactionDate
1	1	100001	Transfer Out	100.00	2024-07-29 14:37:18.890

	TransactionID	AccountNumber	TransactionType	Amount	TransactionDate
1	2	100004	Transfer In	100.00	2024-07-29 14:37:18.893

6. Viewing Transaction History:
    a. Ensure transactions are listed in descending order of date.
    b. Verify all transaction details are correct.

```
EXEC ViewTransactionHistory @AccountNumber = 100001;
EXEC ViewTransactionHistory @AccountNumber = 100004;
```

146 %

Results | Messages

	TransactionID	TransactionType	Amount	TransactionDate
1	6	Withdrawal	150.00	2024-07-29 14:51:52.143
2	5	Deposit	200.00	2024-07-29 14:48:26.123
3	4	Deposit	200.00	2024-07-29 14:48:08.650
4	1	Transfer Out	100.00	2024-07-29 14:37:18.890

	TransactionID	TransactionType	Amount	TransactionDate
1	2	Transfer In	100.00	2024-07-29 14:37:18.893