Bank Management System (SQL Project Documentation)

Developer: Kritika Lohomi

CSI ID: CT_CSI_SQ_4357

Table of Contents

1. Project Overview

This project demonstrates the core backend logic of a banking management system using SQL Server. The project covers:
- Schema design using normalized relational tables
- Implementation of business logic via stored procedures
- Enforcement of data integrity using constraints and validations
- Features such as account creation, money deposit/withdrawal, fund transfer, employee management, and transaction history tracking

2. ER Diagram

The ER Diagram outlines the relationships between core entities such as Person, Customer, Employee, Account, Branch, and Transaction. Each Customer or Employee is a Person. A Customer owns accounts which are linked to a Branch and tracked via Transactions.

3. Database Schema

Person

PersonID INT PRIMARY KEY
FirstName, LastName, DOB, Email, PhoneNumber, Address
TaxIdentifier, AadhaarNumber, PANNumber

Customer

CustomerID INT PRIMARY KEY

PersonID (FK), CustomerType

AadhaarNumber (masked), PANNumber (masked)

Employee

EmployeeID INT PRIMARY KEY

PersonID (FK), Position, Salary

Branch

BranchID INT PRIMARY KEY

BranchName, BranchCode, Address, PhoneNumber

Account

AccountID INT PRIMARY KEY

CustomerID (FK), BranchID (FK)

AccountNumber (unique), AccountType

CurrentBalance, DateOpened, DateClosed, AccountStatus

Transaction

TransactionID INT PRIMARY KEY

AccountID (FK), TransactionType, Amount, TransactionDate

4. Stored Procedures

proc_CreateNewAccount

Creates a new customer and account with validation and returns generated IDs.

proc_DepositAmount

Deposits funds into a given account with balance update and transaction logging.

proc_WithdrawalAmount

Withdraws funds after checking balance, updates account and logs transaction.

proc_TransferAmount

Transfers funds between two accounts, with full transaction logging.

proc_CloseAccount

Marks an account as closed and logs closure.

proc_CreateNewEmployee

Registers a new employee with PAN/Aadhaar validation.

proc_ViewTransactionHistory

Displays customer info and last 10 transactions.

5. Data Integrity and Constraints

- Primary & Foreign Keys: Ensure relational integrity.
- NOT NULL Constraints: Mandatory fields.
- CHECK Constraints: Balance ≥ 0, valid email, transaction amount ≥ 0.
- UNIQUE Constraints: Aadhaar, PAN, AccountNumber.
- DEFAULT Values: AccountStatus ('Active'), TransactionDate (GETDATE()).
- Dynamic Data Masking: Aadhaar → XXXX-XXXX-1234, PAN → XXXX123456.

6. Sample Data & Execution

Sample Account Creation:
EXEC proc_createnewaccount @FirstName = 'Hema', ..., @AccountType = 'Savings';

Deposit:
EXEC proc_DepositAmount @Amount=1200, @AccountID=2;

Withdrawal:
EXEC proc_withdrawalAmount @Amount=1000, @AccountID=3;

Transfer:
EXEC proc_TransferAmount @FromAccountID=1, @ToAccountID=2, @Amount=500;

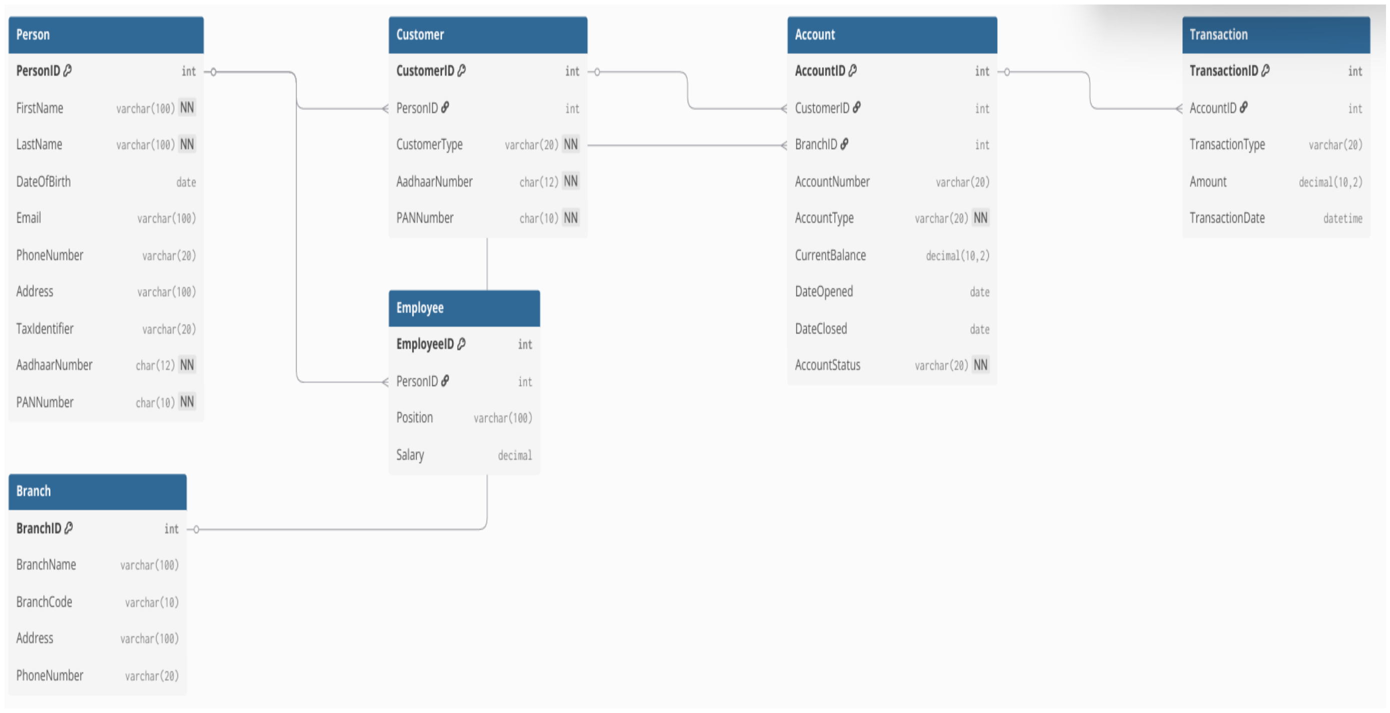
View Transactions:
EXEC proc_ViewTransactionHistory @AccountID=1;

7. Conclusion

This banking system simulation demonstrates a real-world relational design, following best practices of:

- Modularity via stored procedures
- Security-first design (masking, validations)
- Scalable and normalized schema
- Focus on data consistency and auditability

By deploying this system to SQL Server or Azure SQL, it can be further enhanced with triggers, backups, and integration with frontends.


Person
PersonID int
FirstName varchar(100) NN
LastName varchar(100) NN
DateOfBirth date
Email varchar(100)
PhoneNumber varchar(20)
Address varchar(100)
TaxIdentifier varchar(20)
AadhaarNumber char(12) NN
PANNumber char(10) NN
Customer
CustomerID int
PersonID int
CustomerType varchar(20) NN
AadhaarNumber char(12) NN
PANNumber char(10) NN
Employee
EmployeeID int
PersonID int
Position varchar(100)
Salary decimal
Account
AccountID int
CustomerID int
BranchID int
AccountNumber varchar(20)
AccountType varchar(20) NN
CurrentBalance decimal(10,2)
DateOpened date
DateClosed date
AccountStatus varchar(20) NN
Transaction
TransactionID int
AccountID int
TransactionType varchar(20)
Amount decimal(10,2)
TransactionDate datetime
Branch
BranchID int
BranchName varchar(100)
BranchCode varchar(10)
Address varchar(100)
PhoneNumber varchar(20)