Lab 4

Banking System - Project Report

Course: DBI202 | Instructor: Ms. Nguyễn Thị Thu Thảo

Group Members:

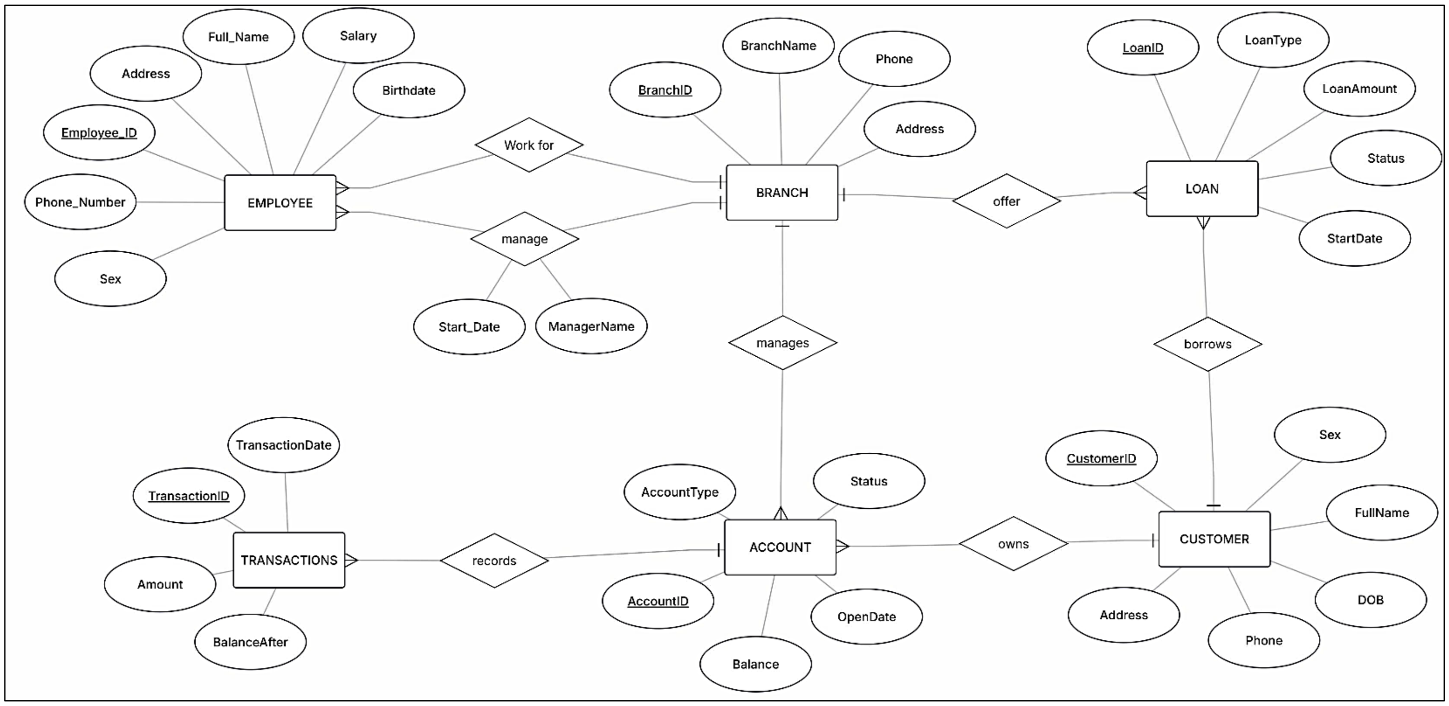
Student ID	**Full Name**	**Group**	**Group Mark**	**Contribution (%)**	**Mark**	**Note**
SE203334	Hà Nguyễn Tiến Đạt	7		100		
SE190596	Trần Hữu Việt	7		100		
SE193659	Mai Thành Được	7		100		
SS190849	Lê Nguyên Ngọc	7		100		

1. Objective

The objective of this lab is to guide the group through the structured process of relational database design for our Banking System. We will practice translating our real-world requirements into a conceptual model (ERD), refining it into a logical model, and implementing it as a physical database schema. Additionally, this report will specify the constraints required to ensure data integrity within the designed schema.

2. Entity–Relationship Diagram (ERD)

The conceptual model was developed to identify the core components of the banking system. This ERD clearly identifies the system's entities, their attributes, and the relationships between them, including cardinalities.



ERD Description:

Entities: EMPLOYEE, BRANCH, CUSTOMER, ACCOUNT, LOAN, TRANSACTIONS.

Attributes:

- **EMPLOYEE**: Employee_ID, Full_Name, Address, Phone_Number, Sex, Birthdate, Salary
- **BRANCH**: BranchID, BranchName, Phone, Address
- **CUSTOMER**: CustomerID, FullName, DOB, Address, Phone, Sex
- **ACCOUNT**: AccountID, AccountType, Balance, Status, OpenDate
- **LOAN**: LoanID, LoanType, LoanAmount, StartDate, Status
- **TRANSACTIONS**: TransactionID, TransactionDate, Amount, BalanceAfter

Relationships & Cardinalities:

- An EMPLOYEE *work for* one BRANCH (N:1).
- An EMPLOYEE *manage* one BRANCH (1:1).
- A BRANCH *manages* many ACCOUNTS (1:N).
- A BRANCH *offer* many LOANS (1:N).
- A CUSTOMER *owns* many ACCOUNTS (1:N).
- A CUSTOMER *borrows* many LOANS (1:N).
- An ACCOUNT *records* many TRANSACTIONS (1:N).

3. Logical Diagram (Relational Schema)

The ERD was converted into a relational schema (Logical Diagram). This process involved defining primary and foreign keys to establish the links between tables. The 1:N relationships were implemented by adding foreign keys to the "many" side of the relationship.

Relational Schema:

EMPLOYEE (Employee_ID, Full_Name, Address, Phone_Number, Sex, Birthdate, Salary, BranchID, ManagedBranchID, ManagerStartDate)

- BranchID (FK) → BRANCH.BranchID
- ManagedBranchID (FK) → BRANCH.BranchID

BRANCH (BranchID, BranchName, Phone, Address)

CUSTOMER (CustomerID, FullName, DOB, Address, Phone, Sex)

ACCOUNT (AccountID, AccountType, Balance, Status, OpenDate, BranchID, CustomerID)

- BranchID (FK) → BRANCH.BranchID
- CustomerID (FK) → CUSTOMER.CustomerID

LOAN (LoanID, LoanType, LoanAmount, StartDate, Status, BranchID, CustomerID)

- BranchID (FK) → BRANCH.BranchID
- CustomerID (FK) → CUSTOMER.CustomerID

TRANSACTIONS (TransactionID, TransactionDate, Amount, BalanceAfter, AccountID)

- AccountID (FK) → ACCOUNT.AccountID

4. Physical Diagram (Detailed Table Design)

The logical schema was mapped to a physical model, defining specific data types and properties for implementation in a SQL database.

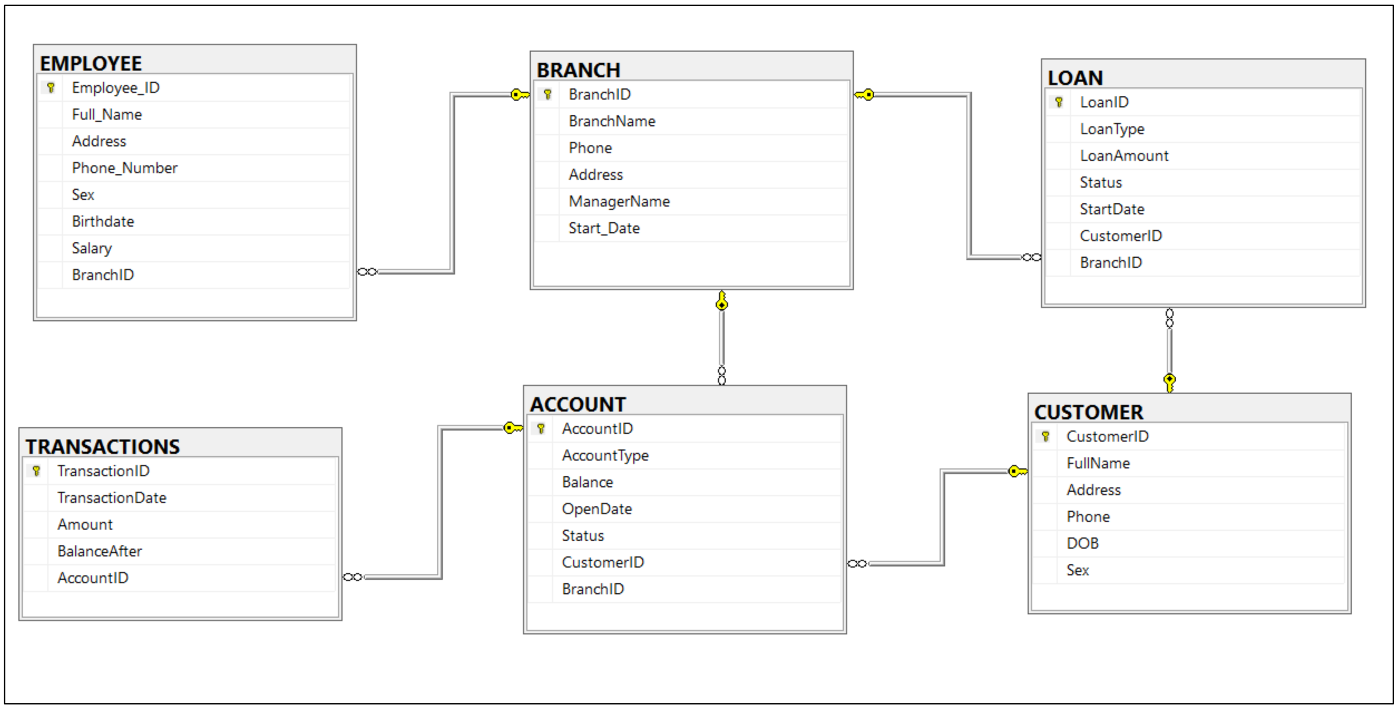


Table Definitions:

```
CREATE TABLE CUSTOMER (
    CustomerID INT PRIMARY KEY,
    FullName VARCHAR(255) NOT NULL,
    Address VARCHAR(255),
    Phone VARCHAR(20) NOT NULL UNIQUE,
    DOB DATE,
    Sex VARCHAR(10) CHECK (Sex IN ('Male', 'Female')) NOT NULL
);

CREATE TABLE BRANCH (
    BranchID INT PRIMARY KEY,
    BranchName VARCHAR(255) NOT NULL UNIQUE,
    Phone VARCHAR(20),
    Address VARCHAR(255),
    ManagerName VARCHAR(255),
    Start_Date DATE
);

CREATE TABLE EMPLOYEE (
    Employee_ID INT PRIMARY KEY,
    Full_Name VARCHAR(255) NOT NULL,
    Address VARCHAR(255),
    Phone_Number VARCHAR(20) NOT NULL UNIQUE,
    Sex VARCHAR(10) CHECK (Sex IN ('Male', 'Female')),
    Birthdate DATE,
    Salary DECIMAL(10,2) NOT NULL CHECK (Salary > 0),
    BranchID INT NOT NULL,
    FOREIGN KEY (BranchID) REFERENCES BRANCH(BranchID)
);

CREATE TABLE ACCOUNT (
    AccountID INT PRIMARY KEY,
    AccountType VARCHAR(100) NOT NULL,
    Balance DECIMAL(12,2) NOT NULL CHECK (Balance >= 0) DEFAULT 0,
    OpenDate DATE NOT NULL,
    Status VARCHAR(50) NOT NULL CHECK (Status IN ('Active', 'Closed', 'Frozen')),
    CustomerID INT NOT NULL,
    BranchID INT NOT NULL,
    FOREIGN KEY (CustomerID) REFERENCES CUSTOMER(CustomerID),
    FOREIGN KEY (BranchID) REFERENCES BRANCH(BranchID)
);

CREATE TABLE LOAN (
    LoanID INT PRIMARY KEY,
    LoanType VARCHAR(100) NOT NULL,
    LoanAmount DECIMAL(12,2) NOT NULL CHECK (LoanAmount > 0),
    Status VARCHAR(50) NOT NULL CHECK (Status IN ('Pending', 'Approved', 'Paid Off', 'Default')),
    StartDate DATE NOT NULL,
    CustomerID INT NOT NULL,
    BranchID INT NOT NULL,
    FOREIGN KEY (CustomerID) REFERENCES CUSTOMER(CustomerID),
    FOREIGN KEY (BranchID) REFERENCES BRANCH(BranchID)
);

CREATE TABLE TRANSACTIONS (
    TransactionID INT PRIMARY KEY,
    TransactionDate DATETIME NOT NULL DEFAULT GETDATE(),
    Amount DECIMAL(12,2) NOT NULL CHECK (Amount <> 0),
    BalanceAfter DECIMAL(12,2),
    AccountID INT NOT NULL,
    FOREIGN KEY (AccountID) REFERENCES ACCOUNT(AccountID)
);
```

5. List and Description of Constraints

To ensure data integrity, the following column-level and table-level constraints were specified.

1. CUSTOMER

- **PRIMARY KEY:** CustomerID
- **NOT NULL:** CustomerID, FullName, Phone
- **UNIQUE:** Phone
- **CHECK:** Sex IN ('Male', 'Female')

2. BRANCH

- **PRIMARY KEY:** BranchID
- **NOT NULL:** BranchID, BranchName
- **UNIQUE:** BranchName

3. EMPLOYEE

- **PRIMARY KEY:** Employee_ID
- **NOT NULL:** Employee_ID, Full_Name, Phone_Number, Salary
- **UNIQUE:** Phone_Number
- **FOREIGN KEY:** BranchID REFERENCES BRANCH(BranchID)
- **CHECK:** Salary > 0

4. ACCOUNT

- **PRIMARY KEY:** AccountID
- **NOT NULL:** AccountID, AccountType, Balance, OpenDate, Status, CustomerID, BranchID
- **FOREIGN KEY:** BranchID REFERENCES BRANCH(BranchID)
- **FOREIGN KEY:** CustomerID REFERENCES CUSTOMER(CustomerID)
- **CHECK:** Balance >= 0
- **DEFAULT:** Balance DEFAULT 0
- **CHECK:** Status IN ('Active', 'Closed', 'Frozen')

5. LOAN

- **PRIMARY KEY:** LoanID
- **NOT NULL:** LoanID, LoanType, LoanAmount, Status, StartDate, CustomerID, BranchID
- **FOREIGN KEY:** BranchID REFERENCES BRANCH(BranchID)
- **FOREIGN KEY:** CustomerID REFERENCES CUSTOMER(CustomerID)
- **CHECK:** LoanAmount > 0
- **CHECK:** Status IN ('Pending', 'Approved', 'Paid Off', 'Defaulted')

6. TRANSACTIONS

- **PRIMARY KEY:** TransactionID
- **NOT NULL:** TransactionID, TransactionDate, Amount, AccountID
- **FOREIGN KEY:** AccountID REFERENCES ACCOUNT(AccountID)
- **DEFAULT:** TransactionDate DEFAULT GETDATE()
- **CHECK:** Amount != 0

6. Conclusion and Reflection

This lab guided our group through a structured database design process: analyzing banking system requirements, creating a conceptual ERD, converting it to a relational schema with keys, and developing a physical model with data types and constraints.

Constraints were shown to be vital for data integrity:

- PRIMARY KEY and UNIQUE prevent duplicates.
- FOREIGN KEY maintains referential integrity.
- NOT NULL ensures essential data is recorded.
- CHECK and DEFAULT enforce business rules and data validity.