Lab 1

Banking System - Project Report

Course: DBI202 | Instructor: Ms. Nguyen Thi Thu Thao

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		

Table of Contents

1. Introduction

The aim of this assignment is to develop a database system to hold records for customers, accounts, loans, transactions, and employees in a banking organization. The system will be used for managing daily banking operations such as account handling, loan processing, and transaction tracking, as well as to support decision-making, compliance reporting, and customer service.

Our bank is organized into several **branches** which are described by a branch code, branch name, location, and contact information. Each branch is **managed** by a particular employee who acts as the branch manager. We keep track of the start date when that employee began managing the branch.

The bank recruits more **employees** and **arranges** them across different branches based on their roles and expertise. Employees are known by an employee ID, name, position, date of hire, and branch assignment. For compliance purposes, we also maintain records of employee actions through an audit log to ensure accountability and transparency.

Banking services are **distributed** among branches. Each branch provides multiple services, including customer account management and **loan** issuance. Account information includes account number, account type (savings, current, fixed deposit), account balance, account status, and the **customer** who owns or shares the account. A branch can manage many accounts, and an account may be jointly held by multiple customers.

Customer information is **critical** for the bank. Each customer has a unique customer ID, name, date of birth, address, phone number, and Know Your Customer (KYC) status. Customers may open multiple accounts across branches and may also apply for different types of loans.

Loan records are also maintained in the system. Information on loans includes loan ID, loan type, principal amount, interest rate, repayment term, and loan status. A branch can issue many loans, and a loan can be jointly availed by multiple customers as co-applicants.

Transactions are recorded for every account. Information on transactions includes transaction ID, type (deposit, withdrawal, transfer, payment), source account, destination account, amount, timestamp, and status. Each account may generate many transactions, and transactions are linked to audit logs for tracking.

The bank also provides additional services such as cards and notifications. Card information includes card ID, type (debit, credit), expiry date, and the account it is **linked** to. Notifications are sent to customers for activities such as account updates, loan reminders, and transaction alerts.

The overall goal of this Banking Management System is to ensure that all customer, account, transaction, loan, and employee records are **maintained** accurately. The system supports security, reliability, and scalability, which are essential for **financial institutions**, while providing effective tools for customer service and regulatory compliance.

2. Project Objectives

- Manage customers, accounts, and transactions (deposit, withdrawal, transfer).
- Manage cards, loans, notifications, and reports.
- Provide secure login, multi-factor authentication, and audit logging.

3. Overview of Data Models

3.1. Structured Data Model (RDBMS – Relational Database Management System)

- **Storage structure:** Data is stored in **tables (relations)** with rows (tuples) and columns (attributes). Primary keys and foreign keys ensure data integrity.
- **Constraints:** Strong constraints (PK, FK, UNIQUE, NOT NULL, CHECK) enforce data validity. **ACID compliance** guarantees reliable transactions.
- **Data manipulation operators:** SQL (SELECT, INSERT, UPDATE, DELETE), joins, aggregation.
- **Advantages:**
  - High consistency and reliability.
  - Transaction management with rollback/recovery.
  - Standardized query language (SQL).
- **Limitations:**
  - Less flexible with unstructured or semi-structured data.
  - Schema changes are costly.
- **Applications:** Financial systems, banking transactions, ERP systems.

3.2. Semi-Structured Data Model (JSON / NoSQL – Document Stores)

- **Storage structure:** Data stored as **documents (JSON, BSON, XML)**, not requiring fixed schema.
- **Constraints:** Weaker than RDBMS; validation rules possible but less strict.
- **Data manipulation operators:** CRUD APIs, flexible queries, indexing on fields.
- **Advantages:**
  - Flexible schema, easy to evolve with changing business needs.
  - Good performance for customer profile storage and personalization.
  - Horizontal scalability (sharding, replication).
- **Limitations:**
  - Weaker consistency (CAP theorem – often eventual consistency).
  - No standard query language across all NoSQL databases.
- **Applications:** Customer information systems, metadata storage, product catalogs.

3.3. Unstructured Data Model (Blob / Object Storage)

- **Storage structure:** Data stored as **binary large objects (blobs)** such as images, PDFs, audio, or video files.
- **Constraints:** Minimal – metadata tagging possible, but no relational constraints.
- **Data manipulation operators:** APIs to upload, retrieve, or delete objects.
- **Advantages:**
    - Best suited for large files and documents.
    - Scalable, low-cost storage.
- **Limitations:**
    - Cannot run SQL queries directly on objects.
    - Requires additional indexing/search systems.
- **Applications:** Storing **KYC documents, identity proofs, customer agreements, bank statements**.

3.4. Graph Data Model (Graph DB)

- **Storage structure:** Data stored as **nodes (entities)** and **edges (relationships)** with properties.
- **Constraints:** Relationship rules can be enforced; graph traversal constraints possible.
- **Data manipulation operators:** Graph query languages (e.g., Cypher) for pattern matching.
- **Advantages:**
    - Excellent for relationship-heavy queries.
    - Enables fraud detection, social network analysis, recommendation systems.
- **Limitations:**
    - Not optimized for heavy transactional workloads.
    - Smaller ecosystem compared to SQL/NoSQL.
- **Applications:** Fraud detection, customer relationship analysis, money-laundering detection.

3.5. Chosen Model: Hybrid Approach

Our group selects a **hybrid model** that combines different data models, leveraging their individual strengths:

- RDBMS (Structured) for **core financial transactions**:

- Ensures **ACID properties** → no lost or duplicate transactions.
- Example: Deposits, withdrawals, transfers, balance updates.

- NoSQL / JSON (Semi-structured) for **customer profiles & dynamic data**:

- Customer data varies (different fields for individuals vs. corporations).
- JSON allows **flexible schema evolution** without downtime.

- Object Storage (Unstructured) for **document management**:

- Store scanned KYC documents, ID cards, bank statements.
- Cost-effective, scalable, secure.

- Graph DB for **fraud analysis & relationship discovery**:

- Detects suspicious account linkages, money-laundering rings.
- Efficient graph traversal for **network anomaly detection**.

3.5.1. Why Hybrid is Best for Banking System?

There are various reasons why Hybrid is Best for Banking System:

- **Reliability + Flexibility:** RDBMS guarantees transaction integrity, while NoSQL provides agility for evolving customer needs.
- **Compliance:** Object storage securely handles regulatory documents (KYC/AML).
- **Fraud Prevention:** Graph DB adds **intelligence** to fight fraud and improve risk management.
- **Scalability:** Hybrid design ensures the system can grow with both structured and unstructured data demands.

4. Computational Thinking Approach

Computational thinking (**CT**) refers to the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms. In education, CT is a set of problem-solving methods that involve expressing problems and their solutions in ways that a computer could also execute. It involves automation of processes, but also using computing to explore, analyze, and understand processes (natural and artificial).

The Banking Management System was analyzed using CT principles to simplify complexity, identify patterns, and design a reliable data model.

Decomposition:

The system is divided into smaller, manageable modules:

- **Customer Management** – storing customer information.
- **Account Management** – managing account types and balances.
- **Transaction Management** – handling deposits, withdrawals, and transfers.
- **Loan Management** – managing loan applications, payments, and status.
- **Employee Management** – managing staff assignments and customer services.

5. AI Utilization

- **Fraud detection:** ML models analyze transactions in real-time.
- **Chatbot:** Provide 24/7 customer support.
- **Recommendation:** Suggest financial products.
- **KYC automation:** OCR + AI for document verification.

6. Teamwork & Collaboration

The project was completed collaboratively. Roles were divided as follows:

1. Hà Nguyễn Tiến Đạt: Database Design & Core Schema

Role: Design and implement the overall database schema.

Details:

- Define tables (**Customer, Account, Transaction, Loan, Employee, Card, Notification**).
- Set up **primary keys, foreign keys, and constraints**.
- Ensure **data normalization** and integrity.
- Provide **ERD diagram** and schema documentation.

2. Trần Hữu Việt: Customer & Account Management

Role: Build customer and account management modules.

Details:

- Implement APIs for **add/edit/delete customers**.
- Implement APIs for **create/update accounts** (savings, current, fixed deposit).
- Design a **basic user interface** for customer/account operations.
- Ensure account status (active, closed, frozen) is properly managed.

3. Mai Thành Được: Transaction Management

Role: Develop transaction handling functions.

Details:

- Build APIs for **deposit, withdrawal, transfer, and payments**.
- Ensure **transaction ACID compliance** and proper error handling.
- Link transactions to **audit log** for traceability.
- Generate **transaction receipts** for customer confirmation.

4. Lê Nguyên Ngọc: Loan, Security & Reporting

Role: Handle loan operations, security, and reporting features.

Details:

- Implement APIs for **loan applications, repayments, and loan status tracking**.
- Develop **secure login system** (username/password + optional MFA).
- Maintain **audit logging** for employee/customer actions.
- Create **reports** (e.g., transaction history, loan status, customer summaries).