**Lab 2**

**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. Scenario

This lab focuses on analyzing the core components of a Banking System database such as entities, attributes, functional dependencies, and keys. Through this exercise, our group aims to understand how to model real-world data into structured entities, define relationships and functional dependencies, and identify keys that ensure data integrity in a database.

**Business Rules**

* Bank have Customer.
* Banks are identified by a *name*, *code*, *address of main office*.
* Banks have branches.
* Branches are identified by a *branch\_no*, *branch\_name*, *address*.
* Customers are identified by *name*, *cust-id*, *phone number*, *address*.
* Customer can have one or more accounts.
* Accounts are identified by *account\_no*, *acc\_type*, *balance*.
* Customer can avail loans.
* Loans are identified by *loan\_id*, *loan\_type* and *amount*.
* Account and loans are related to bank's branch.

**2. Entities and Attributes:**

+ Entities:

* **Bank**: Attributes of Bank Entity are Bank Name, Code and Address.   
  *Code is Primary Key* for Bank Entity.
* **Customer**: Attributes of Customer Entity are Customer\_id, Name, Phone Number and Address.  
  *Customer\_id is Primary Key* for Customer Entity.
* **Branch**: Attributes of Branch Entity are Branch\_id, Name and Address.   
  *Branch\_id is Primary Key* for Branch Entity.
* **Account**: Attributes of Account Entity are Account\_number, Account\_Type and Balance.   
  *Account\_number is Primary Key* for Account Entity.
* **Loan**: Attributes of Loan Entity are Loan\_id, Loan\_Type and Amount.   
  *Loan\_id is Primary Key* for Loan Entity.

+ Relationships & Cardinality:

* Bank has Branch: **1–N**

*One bank has many branches but one branch can not belong to many banks, so the relationship between Bank and Branch is one to many relationship.*

* Branch manages Account: **1–N**

*One branch manages many accounts but one account can not belong to many branches, so the relationship between Branch and Account is one to many relationship.*

* Branch offer Loan: **1–N**

*One branch can have many loans but one loan can not belong to many branches, so the relationship between Branch and Loan is one to many relationship.*

* Account held by Customers: **M–N**

*One customer can have more than one accounts and also one account can be held by one or more customers, so the relationship between Account and Customers is many to many relationship.*

* Loan availed by Customer: **M–N** *(Assume loan can be jointly held by many Customers)*

*One customer can have more than one loans and also one loan can be availed by one or more customers, so the relationship between Loan and customers is many to many relationship.*

+ Relational Diagram:

* **Bank** (Code, Name, Address)
* **Branch** (Branch\_id, Name, Address)
* **Customer** (CusID, Name, Phone, Address)
* **Account** (Account\_No, Acc\_Type, Balance)
* **Loan** (Loan\_id, Loan\_type, Amount)

1. **Functional Dependencies (FDs):**

* **Bank**  
  BankCode → BankName, Address  
  ***(****The bank code uniquely determines the bank name and address)*
* **Branch**  
  BranchID → BankCode, BranchName, Address  
  *(The branch ID uniquely determines the branch name, address, and the bank it belongs to)*
* **Customer**  
  CustomerID → Name, Phone, Address  
  *(The customer ID uniquely determines the customer's name, phone number, and address)*
* **Account**  
  AccountNumber → Type, Balance, BranchID  
  *(The account number uniquely determines the account type, balance, and the branch managing it)*
* **Loan**  
  LoanID → Type, Amount, BranchID  
  *(The loan ID uniquely determines the loan type, amount, and the branch that issued it)*

**Pattern Recognition Process**

To identify functional dependencies, our team carefully examined relationships among attributes within each entity. We looked for attributes that are uniquely determined by others (for example, a unique identifier such as Bank Code or Branch ID). By comparing sample data and observing which attributes always repeated together, we recognized patterns such as:

- Each Bank Code consistently determined a specific Bank Name and Address.

- Each Branch ID always appeared with the same Branch Name and Bank Code.

- Each Customer ID uniquely determined the customer’s personal information.

These patterns allowed us to confidently define the FDs listed above and ensure that the database model accurately reflects real-world constraints.

**Algorithmic Reasoning Steps**

To derive the FDs and keys systematically, our group applied a step-by-step reasoning process:

1. Identify Determinants: We started by finding unique identifiers in each entity *(e.g., Bank Code, Branch ID, Customer ID).*

2. Find Dependent Attributes: Next, we listed the attributes that rely on these identifiers *(e.g., Bank Code → Bank Name, Address)*.

3. Compute Attribute Closure: We checked attribute closures to ensure no redundancy and verified that each determinant could uniquely determine all other attributes in the entity.

4. Determine Candidate Keys: Attributes with full closures were selected as candidate keys.

5. Choose Primary Keys: From the candidate keys, we selected the most suitable primary key based on simplicity and clarity.

This logical process made it easier to derive correct dependencies and ensure normalization consistency.

## ****Key Derivation:****

|  |  |  |
| --- | --- | --- |
| **Entity** | **Candidate Key(s)** | **Primary Key** |
| **Bank** | {Code} | Code |
| **Branch** | {Branch\_id} | Branch\_id |
| **Customer** | {CusID} | CusID |
| **Account** | {Account\_No} | Account\_No |
| **Loan** | {Loan\_id} | Loan\_id |

1. **AI Utilization**

We used AI tools (such as ChatGPT, in study mode) to support our FD discovery and validation process. The AI was prompted with our entity definitions and sample business rules to check whether our identified functional dependencies were consistent with database normalization theory.

For example, AI helped us verify that:

- *BranchID → Code, BranchName, Address* correctly indicates that each branch belongs to one bank.

- *CustomerID → Name, Phone, Address* is a valid FD since the customer ID uniquely identifies customer information.

Additionally, AI provided feedback to refine unclear relationships and confirm that our schema followed 3NF normalization principles. Integrating AI verification improved our confidence in the final relational design.

1. **Conclusion and Reflection:**

By analyzing the banking system, the team designed a relational data model, identifying core entities, attributes, functional dependencies, and primary keys. The challenge of identifying the right relationships helped the team realize that analytical skills and key inference are the foundation for building a well-structured database from real-world scenarios.

All members contributed equally throughout the project. Each member participated in entity identification, FD discovery, and AI-assisted verification. The group discussed and reviewed each decision collectively to ensure accuracy and consistency. During presentation preparation, tasks such as slide design, content explanation, and verbal delivery were divided evenly.