**Lab 3**

**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 identify anomalies within the relational database schema for the Banking System designed in Lab 2. Through this exercise, the group will apply normalization techniques by systematically decomposing relations to produce a well-structured database schema, eliminate anomalies, and achieve higher normal forms.

### 2. Analysis of Anomalies

To analyze anomalies, we consider a hypothetical relation formed by combining information from the Branch and Bank entities, based on the functional dependencies identified in Lab 2: BranchID → BankCode and BankCode → BankName, Address

Assumed Relation: Branch\_Bank\_Info

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| BranchID | BranchName | BranchAddress | BankCode | BankName | BankAddress |
| BR\_HCM\_VCB | VCB District 1 | 132 Ham Nghi, D.1 | VCB | Vietcombank | 198 Tran Quang Khai, Hanoi |
| BR\_HN\_VCB | VCB Hoan Kiem | 56 Ly Thai To | VCB | Vietcombank | 198 Tran Quang Khai, Hanoi |
| BR\_HCM\_ACB | ACB District 3 | 442 NTMK, D.3 | ACB | Asia Commercial Bank | 442 Nguyen Thi Minh Khai, D.3 |

Based on this relation, we have identified the following types of anomalies:

* **Redundancy Anomaly:**
  + **Example:** The information for Vietcombank (BankName and BankAddress) is repeated for every branch associated with it. This wastes storage space and increases the risk of inconsistencies.
* **Update Anomaly:**
  + **Example:** If Vietcombank were to change its head office address, we would need to find and update all records corresponding to every Vietcombank branch. Missing even one record would lead to inconsistent data.
* **Insertion Anomaly:**
  + **Example:** We cannot add a new bank to the system if it does not yet have any branches. This is because the primary key, BranchID, cannot be NULL.
* **Deletion Anomaly:**
  + **Example:** If Asia Commercial Bank (ACB) closes its only branch (BR\_HCM\_ACB), deleting this record would cause the complete loss of all information about the bank itself.

### 3. Identification of Normal Forms

We will determine the normal form for our relations based on the functional dependencies (FDs) provided in the Lab 2 report.

* **Relation: Customer(CustomerID, Name, Phone, Address)**
  + FD: CustomerID → Name, Phone, Address.
  + **Analysis:** All non-key attributes are fully dependent on the primary key CustomerID. There are no transitive dependencies.
  + **Conclusion:** The relation is in **BCNF**.
* **Relations: Account(AccountNumber, Type, Balance) and Loan(LoanID, Type, Amount)**
  + FDs: AccountNumber → Type, Balance and LoanID → Type, Amount.
  + **Analysis:** Similar to Customer, these relations have no partial or transitive dependencies.
  + **Conclusion:** These relations are in **BCNF**.
* **Relation: Branch\_Bank\_Info(BranchID, BranchName, BranchAddress, BankCode, BankName, BankAddress)**
  + Relevant FDs: BranchID → BranchName, BranchAddress, BankCode and BankCode → BankName, BankAddress.
  + **Analysis:**
    1. **1NF:** Satisfied. All attributes contain atomic values.
    2. **2NF:** Satisfied. The primary key BranchID is a single attribute, so no partial dependencies can exist.
    3. **3NF:** **NOT SATISFIED**. A transitive dependency exists: BranchID → BankCode → (BankName, BankAddress). The non-key attributes (BankName, BankAddress) depend on another non-key attribute (BankCode).
    4. **BCNF:** Not satisfied because the relation is not in 3NF.

### 4. Step-by-step Decomposition

To resolve the 3NF violation, we will perform a decomposition of the Branch\_Bank\_Info relation.

* **Original Relation:** Branch\_Bank\_Info(<u>BranchID</u>, BranchName, BranchAddress, BankCode, BankName, BankAddress)
* **The Issue:** A transitive dependency BranchID → BankCode → (BankName, BankAddress) violates 3NF.

**Decomposition Step:** We split the relation into two smaller relations to eliminate the transitive dependency:

1. **Relation 1: Branch**
   * This relation retains the attributes that are directly dependent on the primary key BranchID. BankCode is kept as a foreign key.
   * **Structure:** Branch(<u>BranchID</u>, Name, Address, BankCode)
   * **Preserved FD:** BranchID → Name, Address, BankCode.
2. **Relation 2: Bank**
   * A new relation is created from the attribute causing the transitive dependency (BankCode) and the attributes that depend on it.
   * **Structure:** Bank(<u>BankCode</u>, Name, Address)
   * **Preserved FD:** BankCode → Name, Address.

**Evaluation:** This decomposition process ensures two crucial properties:

* **Lossless Join Property:** We can join the new Branch and Bank relations on BankCode to perfectly reconstruct the original information without any loss.
* **Dependency Preservation Property:** All original functional dependencies are maintained within the new relations.

After decomposition, both new relations are in **BCNF**.

### 5. Final Normalized Schema

The following is the final set of normalized relations, including foreign keys to represent the relationships between tables.

**1. Bank**

* BankCode (Primary Key)
* Name
* Address

**2. Branch**

* BranchID (Primary Key)
* Name
* Address
* BankCode (Foreign Key → Bank.BankCode)

**3. Customer**

* CustomerID (Primary Key)
* Name
* Phone
* Address

**4. Account**

* AccountNumber (Primary Key)
* Type
* Balance
* BranchID (Foreign Key → Branch.BranchID)
* CustomerID (Foreign Key → Customer.CustomerID)

**5. Loan**

* LoanID (Primary Key)
* Type
* Amount
* BranchID (Foreign Key → Branch.BranchID)
* CustomerID (Foreign Key → Customer.CustomerID)

**6. Transaction**

* TransactionID (Primary Key)
* AccountNumber (Foreign Key → Account.AccountNumber)
* Date
* Amount
* Type

**Elimination of Anomalies in the New Schema** **:**

* **Redundancy:** Bank information is stored only once in the Bank table.
* **Update:** To change a bank's address, only a single record in the Bank table needs to be updated.
* **Insertion/Deletion:** Banks and branches can be added or deleted independently without causing data loss for the other.

### 6. Conclusion

This lab allowed the group to successfully apply normalization theory to a practical database schema. By analyzing functional dependencies, we identified and resolved a 3rd Normal Form violation. The decomposition of the original relation into separate Branch and Bank relations has thoroughly eliminated data anomalies, bringing the entire schema into BCNF. The final design ensures data consistency, integrity, and greater efficiency for management and maintenance.