University of Asia Pacific

Computer Science and Engineering

Course Title: Database Systems Lab

Course Code: CSE 212

Project Title: **Bank Management System**

Submitted by:
1. Name : Md. Sakib Hossaine
ID : 22201185
2. Name : Md Tabiur Rahman
ID : 22201171
3. Name : Mehedi Hasan Khan Riyad
ID : 22201172
4. Name : Jannat Ara Simi
ID : 22201178

Submitted to:
Alif Ruslan
Lecturer
University of Asia Pacific

Contents

Bank Management System

This project is a **Bank Management System** implemented using **MySQL**. It efficiently organizes and manages various aspects of a banking institution, such as customer details, accounts, loans, transactions, and credit cards. The system is designed to handle complex banking operations with well-structured entities and relationships, ensuring data consistency and integrity through the use of foreign keys and constraints. This solution facilitates seamless integration of banking operations like account management, loan tracking, transactions, and credit card services.

Description

The **Bank Management System** consists of several interconnected tables that represent core entities and their relationships. The system enables functionalities such as maintaining customer records, managing accounts, issuing loans, processing transactions, and administering credit cards. Additionally, the system supports queries to retrieve important data insights, such as total loan balances per branch, customer transactions, and high-balance accounts. The structured schema and relationships ensure data consistency and provide an accurate representation of a bank's operational data.

Entities:

1. **Branch** – Represents the physical location of the bank. Each branch offers services such as account creation, loan issuance, and credit card handling.
2. **Banker** – Employees assigned to manage customer relationships and oversee operations for specific branches.
3. **Customer** – Individuals who interact with the bank to open accounts, apply for loans, and manage their finances.
4. **Account** – Represents the financial account of a customer, which could be of various types (savings, student, etc.).
5. **Loan** – Financial service provided to customers with a repayment plan. Loans are tied to accounts and branches.
6. **Loan Payment** – Represents the payments made by customers toward their loans.
7. **Transaction** – Details the flow of money in or out of an account (e.g., deposits, withdrawals).
8. **Credit Card** – Issued to customers with a set limit, allowing them to make purchases on credit.

Relationship Sets:

1. **Branch-Account**
   - A branch can have multiple accounts, but each account belongs to one branch. **Relationship:** One-to-Many (1:M).
2. **Branch-Banker**
   - A branch can have multiple bankers, but each banker works at one branch. **Relationship:** One-to-Many (1:M).
3. **Account-Customer**
   - Each account is linked to one customer, but a customer can have multiple accounts. **Relationship:** One-to-Many (1:M).
4. **Branch-Loan**
   - A branch can issue multiple loans, but each loan is managed by one branch. **Relationship:** One-to-Many (1:M).
5. **Account-Loan**
   - Each loan is associated with one account, but an account can have multiple loans. **Relationship:** One-to-Many (1:M).
6. **Loan-Loan Payment**
   - A loan can have multiple payments, but each payment is tied to one loan. **Relationship:** One-to-Many (1:M).
7. **Account-Transaction**
   - An account can have multiple transactions, but each transaction belongs to one account. **Relationship:** One-to-Many (1:M).
8. **Customer-Credit Card**
   - A customer can have multiple credit cards, and each credit card is linked to a specific account. **Relationship:** One-to-Many (1:M).

Attributes:

Branch

- branch_id (Primary Key)
- branch_name
- address

Banker

- banker_id (Primary Key)
- branch_id (Foreign Key referencing branch)
- banker_name

Account

- account_id (Primary Key)
- balance
- account_type (e.g., 'savings', 'student')
- branch_id (Foreign Key referencing branch)

Customer

- customer_id (Primary Key)
- customer_name
- mobileno
- dob
- account_id (Foreign Key referencing account)

Loan

- loan_id (Primary Key)
- issue_date
- remaining_balance
- amount
- loan_limit
- branch_id (Foreign Key referencing branch)
- account_id (Foreign Key referencing account)

Loan Payment

- payment_id (Primary Key)
- loan_id (Foreign Key referencing loan)
- amount

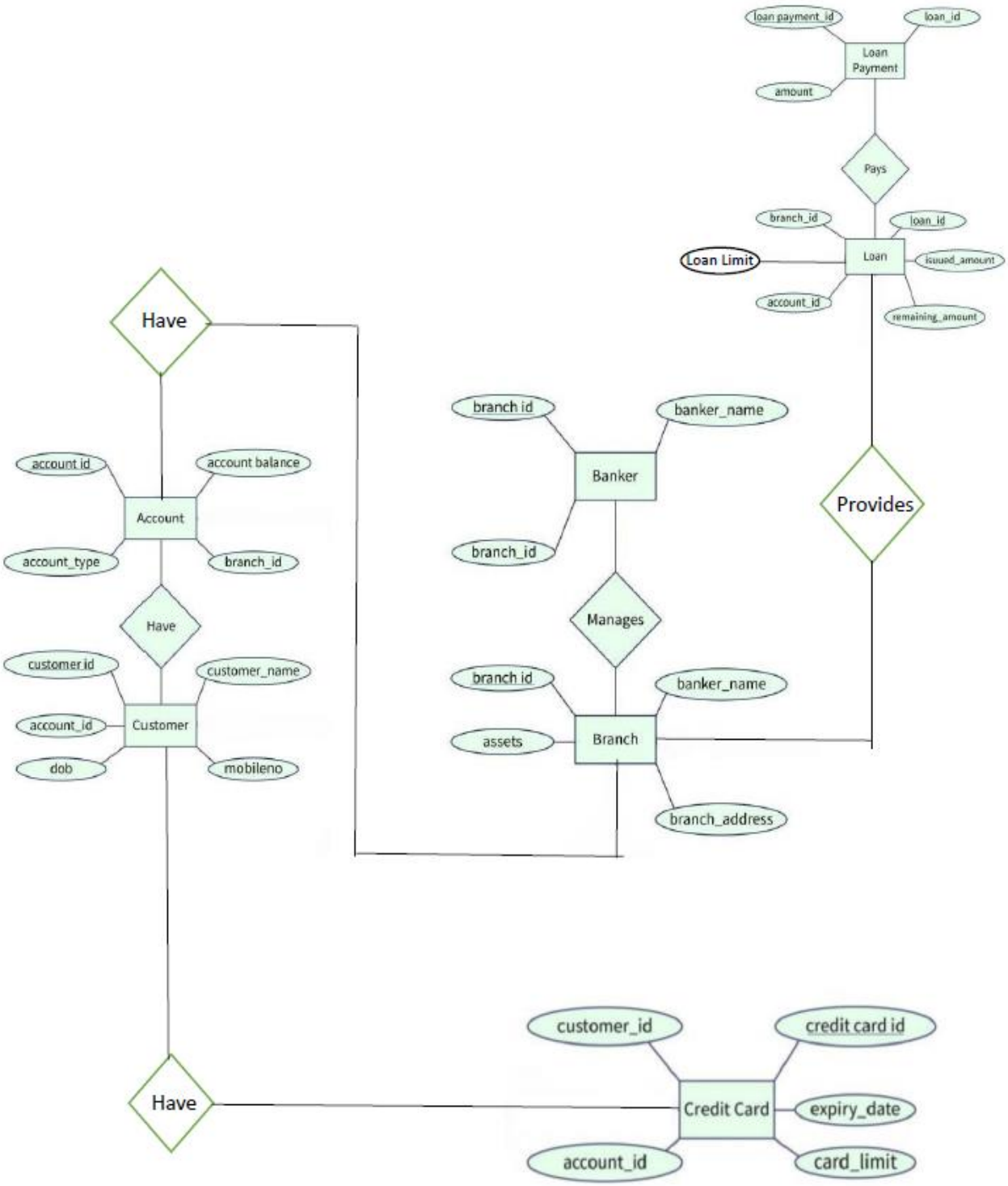
Transaction

- transaction_id (Primary Key)
- account_id (Foreign Key referencing account)
- customer_id (Foreign Key referencing customer)
- amount

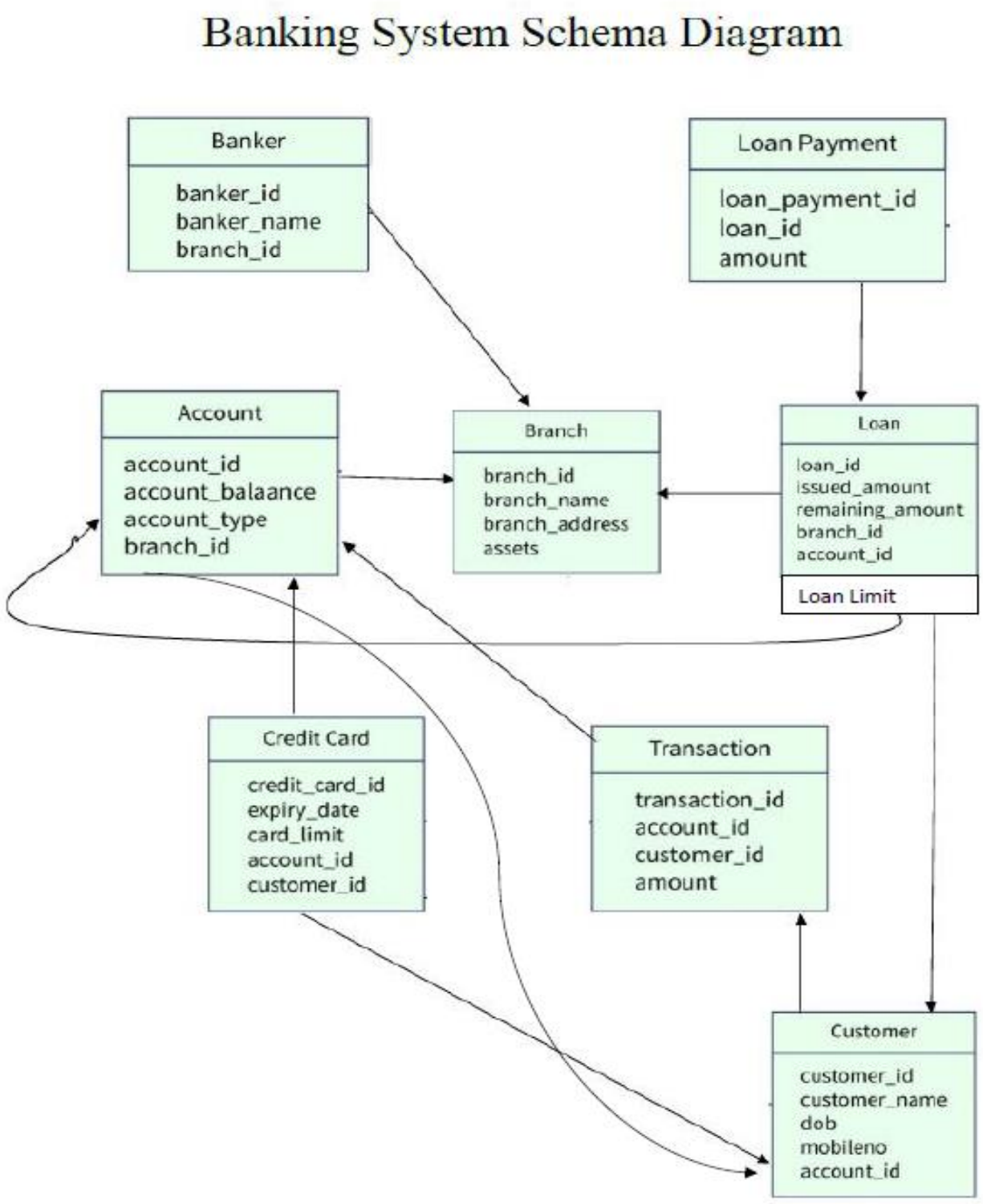
Customer Credit Card

- credit_card_id (Primary Key)
- expiry_date
- card_limit
- customer_id (Foreign Key referencing customer)
- account_id (Foreign Key referencing account)

ER Diagram:


loan payment_id
loan_id
Loan Payment
amount
Pays
branch_id
loan_id
Loan Limit
Loan
issued_amount
account_id
remaining_amount
Have
branch id
banker_name
account id
account balance
Banker
Provides
Account
account_type
branch_id
branch_id
Have
Manages
customer id
customer_name
branch id
banker_name
account_id
Customer
assets
Branch
dob
mobileno
branch_address
customer_id
credit card id
Have
Credit Card
expiry_date
account_id
card_limit

Schema Diagram:



MY SQL QUERYS:

1. Finding Branch table data.

SELECT * FROM branch;

branch_id	branch_name	address
1	Farmgate Branch	123 Dhanmondi
2	Green-road Branch	456 Farmgate
3	Dhanmondi Branch	789 Dhanmondi
4	Gulshan Branch	101 Gulshan

2. Find all customers with a savings account.

```
SELECT customer_name, mobileno, dob
FROM customer
INNER JOIN account ON customer.account_id = account.account_id
WHERE account_type = 'savings';
```

customer_name	mobileno	dob
Sakib	015555555	2000-03-15
rifat panda	0133333333	2002-10-01

3. Count the number of accounts per account type (student or savings).

```
SELECT account_type, COUNT(account_id) AS number_of_accounts
FROM account
GROUP BY account_type;
```

account_type	number_of_accounts
savings	2
student	2

4. Show all bankers working at a specific branch

```
SELECT banker.banker_name
FROM banker
INNER JOIN branch ON banker.branch_id = branch.branch_id
WHERE branch.branch_name = 'Main Branch';
```

Output:

```
Program did not output anything!
```

5. List all branches with the number of bankers working in each

```
SELECT branch.branch_name, COUNT(banker.banker_id) AS number_of_bankers
FROM branch
LEFT JOIN banker ON branch.branch_id = banker.branch_id
GROUP BY branch.branch_name;
```

branch_name	number_of_bankers
Farmgate Branch	1
Green-road Branch	1
Dhanmondi Branch	1
Gulshan Branch	1

6. Calculate the average balance of accounts for each branch

```
SELECT branch.branch_name, AVG(account.balance) AS average_balance
FROM branch
INNER JOIN account ON branch.branch_id = account.branch_id
GROUP BY branch.branch_name;
```

branch_name	average_balance
Farmgate Branch	1000.000000
Green-road Branch	1500.000000
Dhanmondi Branch	2000.000000
Gulshan Branch	2500.000000

7. Retrieve the highest loan amount issued at each branch

SELECT branch.branch_name, MAX(loan.amount) AS highest_loan_amount
FROM loan
INNER JOIN branch ON loan.branch_id = branch.branch_id GROUP BY branch.branch_name;

branch_name	highest_loan_amount
Farmgate Branch	1000.00
Green-road Branch	1500.00
Dhanmondi Branch	2000.00
Gulshan Branch	2500.00

8. Find all credit card holders with their account balances and credit limits

SELECT customer.customer_name, account.balance, customer_credit_card.card_limit
FROM customer
INNER JOIN account ON customer.account_id = account.account_id
INNER JOIN customer_credit_card ON customer.customer_id = customer_credit_card.customer_id;

customer_name	balance	card_limit
Sakib	1000.00	5000
Riyad	1500.00	6000
rifat panda	2000.00	7000
Sajid	2500.00	8000

9. Retrieve the balances of all accounts with a balance greater than 5000. If no such accounts exist, display 'No Related Data

```
SELECT CAST(balance AS CHAR) AS balance
        FROM account   WHERE balance > 5000
  UNION
  SELECT 'No Related Data'   WHERE NOT EXISTS (
    SELECT 1          FROM account
      WHERE balance > 5000          );
```

balance
No Related Data

10. Find all transactions for a specific customer

```
SELECT
transaction.transaction_id, transaction.amount, transaction.account_id
FROM transaction

INNER JOIN customer ON transaction.customer_id = customer.customer_id
WHERE customer.customer_name = 'nonexistent_customer'
UNION
SELECT NULL, 'No Related Data', NULL
WHERE NOT EXISTS (
SELECT 1
FROM transaction
INNER JOIN customer ON transaction.customer_id = customer.customer_id
WHERE customer.customer_name = 'nonexistent_customer'
);
```

```
+----------------+-----------------+------------+
| transaction_id | amount          | account_id |
+----------------+-----------------+------------+
|           NULL | No Related Data |       NULL |
+----------------+-----------------+------------+
```

CEP Mapping:

Ks	Attribute	How Ks are addressed through the project
K2	Mathematics	Used mathematical models for financial projections and optimization of account balances and loan calculations.
K3	Engineering fundamentals	Leveraged database design and SQL query fundamentals to create and manage the banking system.
K4	Specialist knowledge	Integrated advanced database indexing techniques and relational database theory to optimize data retrieval and storage.
K5	Engineering design	Designed a comprehensive schema to meet functional requirements like scalability and consistency in managing banking data.
K6	Engineering practice	Used MySQL on an online compiler website to test, debug, and execute SQL queries, ensuring functionality and correctness.
K7	Comprehension	Demonstrated understanding by solving complex queries such as transactions and loan details with fallback mechanisms.
K8	Research literature	Researched database normalization, data integrity constraints, and best practices for modern financial systems.

How Complex Engineering Problems (Ps) are addressed through the project and mapping among Ps, COs, POs:

Ps	Attribute	How Ps are addressed through the project	COs	POs
P1	Depth of knowledge required	The project requires in-depth knowledge of database fundamentals (K3), E-R diagram & schema design (K5), and SQL to implement the system while considering societal impact.	CO1, CO2, CO3, CO4	PO1, PO3, PO5, PO7
P3	Depth of analysis required	Extensive analysis ensures effective handling of relational data, implementing normalization, and managing multiple stakeholders' data.	CO3, CO4, CO5	PO6, PO7, PO8
Ps	Extent of stakeholder	The system supports diverse stakeholders by accommodating roles like donors, administrators, and beneficiaries for robust interaction.	CO6	PO6, PO8
Ps	Interdependence	The project involves interconnected modules such as donation tracking, user management, and event scheduling for seamless operations.	CO7, CO8	PO11, PO12

How Complex Engineering Activities (As) are addressed through the project and mapping among As, COs, POs:

As	Attribute	How As are addressed through the Banking System project	COs	POs
A1	Range of resources	The Banking System integrates database management tools (MySQL), hardware (personal computer), and human resources to handle transactions and data efficiently.	CO8, CO9	PO11
A4	Consequences for society and environment	The Banking System improves financial transparency, ensures secure online transactions, and reduces manual errors, benefiting society at large.	CO6	PO6, PO12
A5	Familiarity	The Banking System provides a user-friendly interface for customers and bank employees, ensuring smooth interactions for tasks like fund transfers and account management	CO9	PO11, PO12