PROJECT DOCUMENTATION
BANKING SYSTEM DATABASE

Introduction

The Banking System Database project aims to design and implement a robust, scalable, and efficient database system for managing banking operations. This system includes the management of branches, customers, and their respective accounts along with loans and transactions.

The bank is organized into branches, and each branch has a unique identifier, a name and an address. A branch serves an arbitrary number of customers. Each customer has a unique identifier, a first and last name, the date of birth and their gender. A customer also administers one or more bank accounts. A bank account has a unique identifier and its current balance which can also initiate loans and transactions.

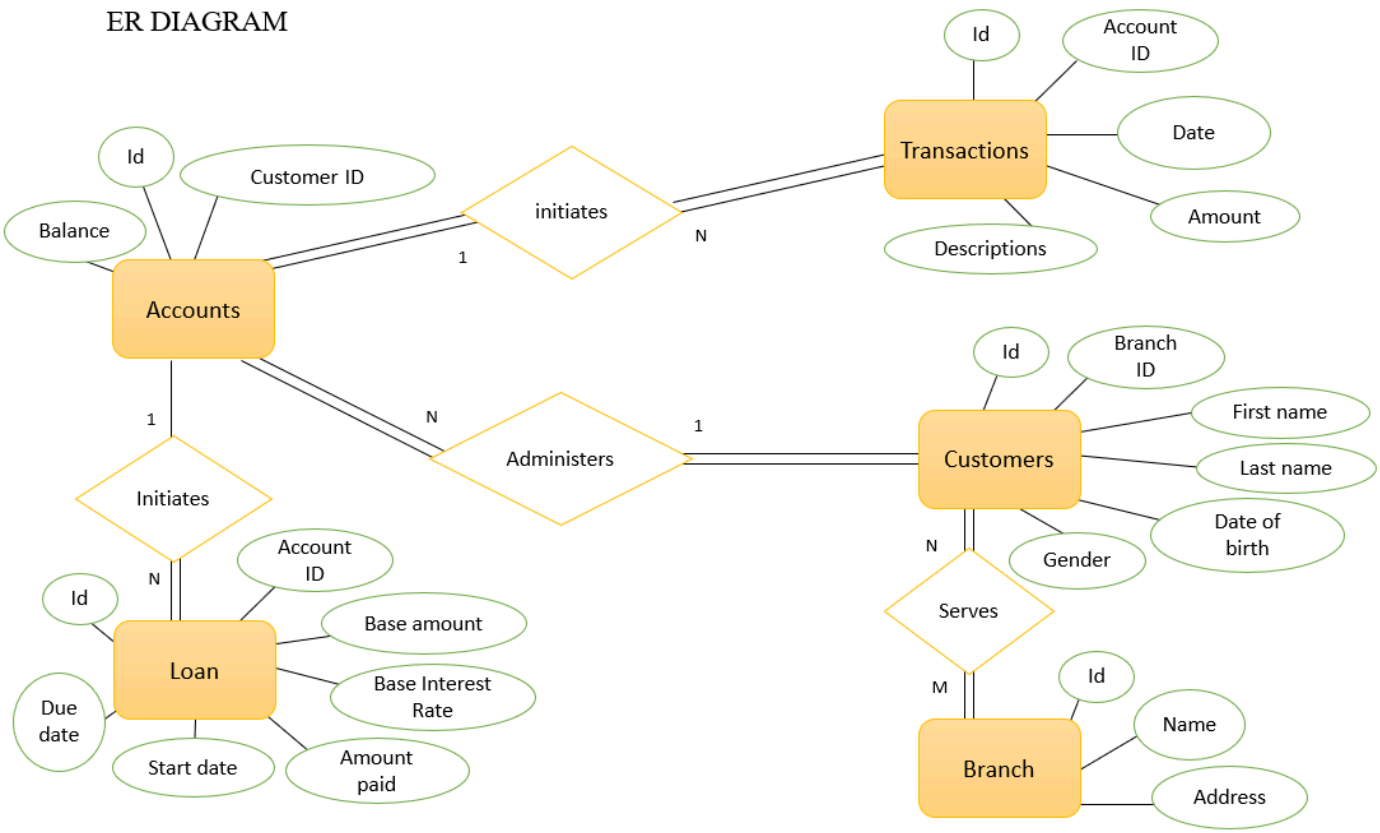
The database keeps track of loans which shows the amount of money that the customer has already paid back and in addition the start and the due dates of the loan. It also gives a base amount and a base interest rate on the particular loan. The database also holds records of transactions, each having a unique identifier, a description of the transaction and the amount and the date of the transfer.

Project Objectives

- Accuracy and Integrity: Ensure high data accuracy and integrity through proper database design and constraint enforcement.
- Capacity and Scalability: Design the database to support a large number of branches and customers, ensuring scalability for future growth.
- Efficiency and Performance: Optimize database performance to enhance query speed and reduce data redundancy.

Database design & ER Diagram

The database schema includes the following core tables:



Branch Table:

- **ID**: Unique identifier for each branch.(Primary Key)
- **Name**: Name of the branch, enforced as unique to prevent duplicates.
- **Address**: Address of the branch.

Customers Table:

- **ID**: Unique identifier for each customer.(Primary Key)
- **Branch_ID**: Foreign key linking the customer to a branch.
- **First_name**: First name of the customer.
- **Last_name**: Last name of the customer.
- **Date_of_birth**: Date of birth of the customer.
- **Gender**: Gender of the customer.(male or female)

Accounts Table:

- **ID**: Unique identifier for each account.(Primary Key)
- **Customer_ID**: Foreign key linking the account to a customer.
- **Balance**: Balance present in the account.

Loan Table:

- **ID**: Unique identifier for each loan.(Primary Key)
- **Account_ID**: Foreign key linking the loan to the account of a customer.
- **Base_amount**: Total loan amount.
- **Amount_paid:** The total amount of loan paid till date.
- **Base_Interest_Rate**: Interest rate applied to the loan.
- **Start_Date**: The date loan was given on.
- **Due_Date**: The last date before which the entire loan must be repaid to the bank.

Transactions Table:

- **ID**: Unique identifier for each transaction.(Primary Key)
- **Account_ID**: Foreign key linking the transaction to an account of a customer in a bank.
- **Transaction_Date**: Date of the transaction.
- **Amount**: Transaction amount.
- **Descriptions**: Gives whether the transaction is successful or not.

Normalization

Normalization was applied to minimize data redundancy and improve data integrity

- **1st form of normalization**: Ensured that all tables have primary keys and atomic values.
- **2nd form of normalization:** Removed partial dependencies by ensuring that non-key attributes are fully functional dependent on the primary key.
- **3rd form of normalization:** Ensured that all non-primary keys are not dependent on another non-primary keys.

Creating Database

Creating a database “bank”

```
CREATE DATABASE Bank;
USE bank;
```

Creating Branch Table:

```
CREATE TABLE Branch(ID INT,
      Name CHAR(50) UNIQUE,
      Address CHAR(50),
      PRIMARY KEY(ID)
);
```

Creating Customers Table:

```
CREATE TABLE Customers(ID INT,
      Branch_ID INT,
      First_name CHAR(20) NOT NULL,
      Last_name CHAR(20) NOT NULL,
      Date_of_birth DATE,
      Gender CHAR(6),
      PRIMARY KEY(ID),
      FOREIGN KEY (Branch_ID) REFERENCES branch(ID) ON UPDATE CASCADE
);
```

Creating Accounts table:

```
CREATE TABLE Accounts(ID INT,
      Customer_ID INT,
      Balance CHAR(20),
      PRIMARY KEY(ID),
      FOREIGN KEY(Customer_ID) REFERENCES Customers(ID) ON UPDATE CASCADE
);
```

Creating Loan table:

```
CREATE TABLE Loan(ID INT,
      Account_ID INT,
      Base_amount DECIMAL(10,3),
      Base_interest_rate DECIMAL(10,3),
      Amount_paid DECIMAL(10,3),
      Start_date DATE,
      Due_date DATE,
      PRIMARY KEY(ID),
      FOREIGN KEY(Account_ID) REFERENCES Accounts(ID) ON UPDATE CASCADE
);
```

Creating Transactions Table:

```
CREATE TABLE Transactions(ID INT,
      Account_ID INT,
      Amount DECIMAL(10,3),
      Descriptions CHAR(100),
      Transaction_date DATE,
      PRIMARY KEY(ID),
      FOREIGN KEY(Account_ID) REFERENCES Accounts(ID) ON UPDATE CASCADE
);
```

Tables after inserting with values

Branch table

ID	Name	Address
1	State Bank of India	Chennai
2	Canara Bank	Bangalore
3	Bank of India	New Delhi
4	Bank of Maharashtra	Mumbai
5	HDFC Bank	Hyderabad
6	ICICI Bank	Bangalore
7	Axis Bank	Pune

OK, 7 records retrieved in 78.292ms

Customers table

ID	Branch_ID	First_name	Last_name	Date_of_birth	Gender
1	1	Laalu	Prasad	10/07/1996	Male
2	3	Rangu	Yadav	09/02/1998	Male
3	1	Samiksha	Naidu	02/17/1997	Female
4	2	Simon	Joseph	04/26/1998	Male
5	2	Ananya	Mishra	10/11/1996	Female
6	6	Sagar	Sharma	05/20/1999	Male
7	5	Pranavi	Desai	03/03/1998	Female

OK, 7 records retrieved in 0s

Accounts table

ID	Customer_ID	Balance
1	1	10000
2	2	100
3	3	2000
4	5	500
5	5	20000
6	6	3000
7	6	200

OK, 7 records retrieved in 7.595ms

Loan table

ID	Account_ID	Base_amount	Base_interest_rate	Amount_paid	Start_date	Due_date
1	1	1000.000	5.000	0.000	02/24/2020	10/24/2024
2	4	20000.000	15.000	0.000	03/04/2021	03/04/2026
3	5	15000.000	3.000	1000.000	10/11/2022	10/11/2024
4	3	100000.000	5.000	20000.000	12/12/2020	12/12/2024
5	2	50000.000	3.500	100.000	04/23/2022	04/23/2025
6	5	20000.000	5.000	0.000	05/14/2023	05/14/2025
7	6	30000.000	7.000	2000.000	09/08/2023	09/08/2026

OK, 7 records retrieved in 10.929ms

Transactions table

ID	Account_ID	Amount	Descriptions	Transaction_date
1	1	1000.900	Success	03/05/2024
2	2	500.000	Success	07/06/2024
3	5	300.500	Success	08/02/2024
4	3	200.000	Fail	07/27/2024
5	4	100.600	Success	07/23/2024
6	1	20.350	Success	06/24/2024
7	7	1000.000	Success	04/19/2024

OK, 7 records retrieved in 9.299ms

Queries

On running, this query shows that men take more loans than women

```
SELECT Gender,COUNT(*) AS count
FROM customers AS c
WHERE c.ID IN(
    SELECT customer_ID
    FROM accounts AS a
    WHERE a.ID IN (
        SELECT Account_ID
        FROM loan AS l))
GROUP BY Gender ORDER BY COUNT DESC;
```

This query shows the customers that have never taken a loan.

```
SELECT c.first_name, c.last_name
    FROM customers AS c
    WHERE c.id IN (SELECT a.customer_id
        FROM accounts AS a
        WHERE a.id NOT IN (SELECT l.account_id
                FROM Loan l));
```

This query shows the customers that have taken more than one loan.

```
SELECT c.first_name, c.last_name
    FROM customers AS c
    WHERE c.id IN (SELECT a.customer_id
        FROM accounts AS a
        WHERE a.id IN (SELECT l.account_id
                FROM Loan l
                GROUP BY l.Account_id
                HAVING COUNT(l.ID)>1));
```

This query shows the customers that have not started paying off their loan.

```
SELECT c.first_name, c.last_name
    FROM customers AS c
    WHERE c.id IN (SELECT a.customer_id
        FROM accounts AS a
```

```
        WHERE a.id IN (SELECT l.account_id
                FROM Loan l
                WHERE l.Amount_paid=0)
                );
```

This query shows the customers that have no open accounts

```
SELECT c.first_name, c.last_name
    FROM customers AS c
    WHERE c.id NOT IN (SELECT customer_id
        FROM accounts AS cb
        GROUP BY customer_id);
```

This query shows the banks that have failed transactions

```
SELECT b.Name,b.Address
    FROM branch AS b
    WHERE b.id IN( SELECT c.branch_id
        FROM customers as c
        WHERE c.id IN( SELECT a.customer_id
            FROM accounts as a
            WHERE a.id IN ( SELECT d.account_id
                FROM transactions as d
                WHERE Descriptions='Fail'
            )));
```

Conclusions

The Banking System Database project successfully achieved its objectives of ensuring data accuracy, integrity, scalability, and performance optimization. Improved query performance by primary keys and foreign keys, reducing the time required for data retrieval and manipulation operations. Separated logically distinct data into different tables, reduced redundancy and improved data integrity.
This database is designed to support extensive scalability upto 10,000 branches and customer records, making it capable of handling large-scale banking operations effectively.