Bank Management System:

Comprehensive Database Architecture & Implementation

1. Introduction

This report presents the design and implementation approach for a comprehensive Bank Management System database. The system is designed to manage a banking environment including Banks, Branches, Employees, Customers, Accounts, Transactions, Loans, Cards, and ATMs. It ensures data accuracy, scalability, and referential integrity for real-world use.

2. Objectives

• To design a complete banking database structure using QuickDBD.
• To establish Primary Keys (PK) and Foreign Keys (FK) for referential integrity.
• To support key banking operations: deposits, withdrawals, fund transfers, loan servicing, and card issuance.
• To export the database schema to MySQL for deployment and querying.

3. Tools and Technologies

• QuickDBD – For creating the Entity Relationship Diagram (ERD) and database schema.
• MySQL – For deploying and running SQL queries on the designed database.
• Microsoft Word – For documentation and report presentation.

4. Using QuickDBD to Design the Database

Step 1: Open QuickDBD and create a new project.
Step 2: Define all entities with their attributes using QuickDBD syntax.
Step 3: Set Primary Keys (PK) using the asterisk (*) notation.
Step 4: Set Foreign Keys (FK) using references between entities.
Step 5: Verify the ERD diagram for correctness and consistency.
Step 6: Export the database schema from QuickDBD to SQL format.

5. System Entities and Attributes

The banking system includes the following main entities with their attributes:

1. Bank:

Bank_ID (PK), Bank_Name, Headquarters, Established_Date

2. Branch:

Branch_ID (PK), Bank_ID (FK), Branch_Name, Address, City, State, Contact

3. Employee:

Employee_ID (PK), Branch_ID (FK), Name, Position, Hire_Date, Salary

4. Customer:

Customer_ID (PK), Name, Address, Phone, Email, Date_of_Birth

5. Account:

Account_ID (PK), Customer_ID (FK), Branch_ID (FK), Account_Type, Balance, Date_Opened

6. Transaction:

Transaction_ID (PK), Account_ID (FK), Transaction_Type, Amount, Date, Description

7. Loan:

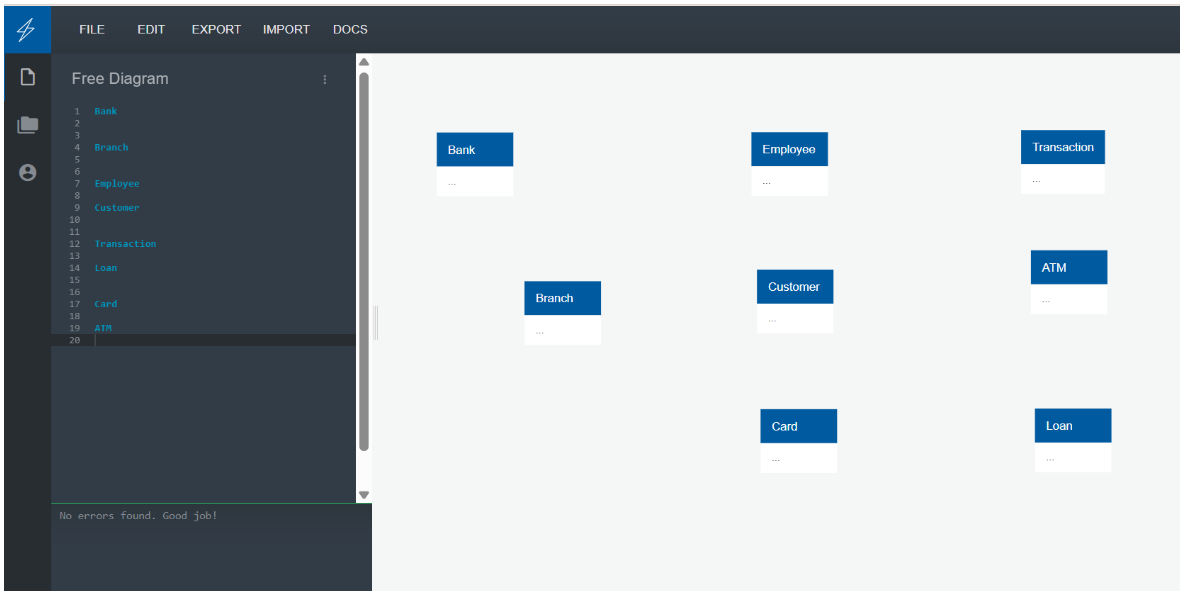
Loan_ID (PK), Customer_ID (FK), Branch_ID (FK), Loan_Type, Amount, Interest_Rate, Start_Date, End_Date

8. Card:

Card_ID (PK), Customer_ID (FK), Account_ID (FK), Card_Type, Issue_Date, Expiry_Date

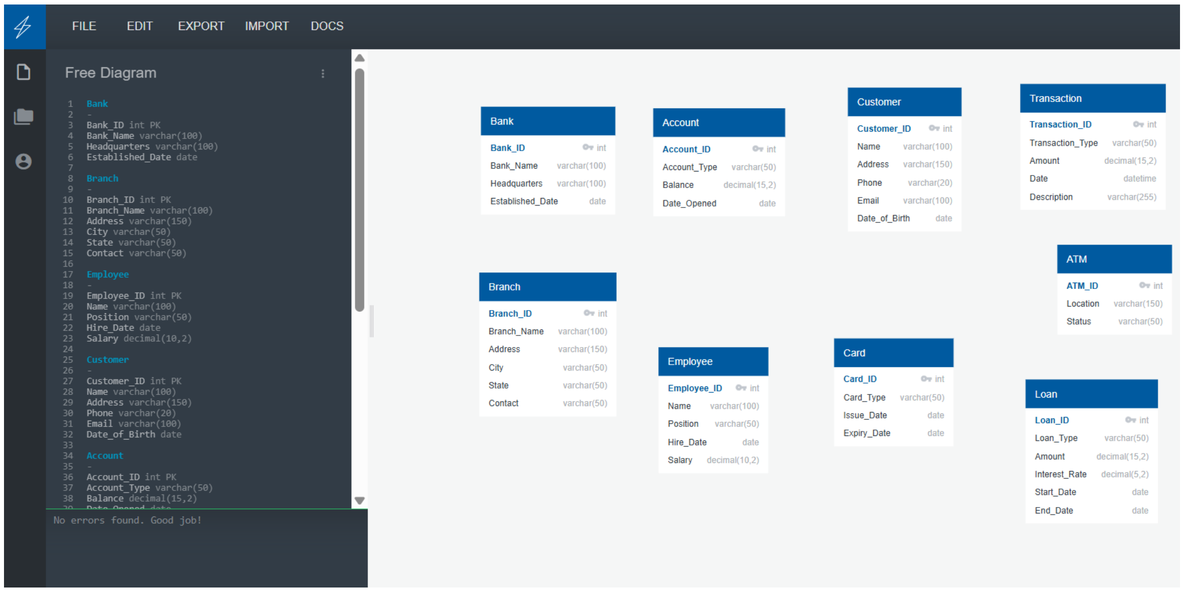
9. ATM:

ATM_ID (PK), Branch_ID (FK), Location, Status



```
Bank


Branch


Employee

Customer


Transaction

Loan


Card

ATM

```

No errors found. Good job!

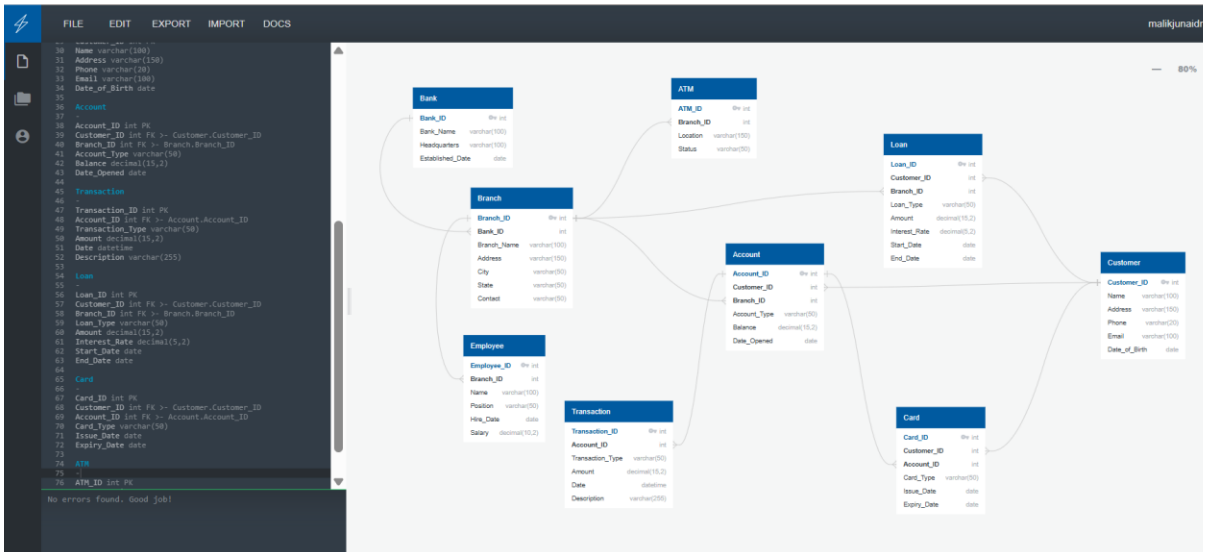


```
Bank
-
Bank_ID int PK
Bank_Name varchar(100)
Headquarters varchar(100)
Established_Date date

Branch
-
Branch_ID int PK
Branch_Name varchar(100)
Address varchar(150)
City varchar(50)
State varchar(50)
Contact varchar(50)

Employee
-
Employee_ID int PK
Name varchar(100)
Position varchar(50)
Hire_Date date
Salary decimal(10,2)

Customer
-
Customer_ID int PK
Name varchar(100)
Address varchar(150)
Phone varchar(20)
Email varchar(100)
Date_of_Birth date

Account
-
Account_ID int PK
Account_Type varchar(50)
Balance decimal(15,2)
```

```
No errors found. Good job!
```

6. Relationships and Referential Integrity

- A Bank can have multiple Branches.
- Each Branch employs multiple Employees.
- A Customer can hold multiple Accounts and Loans.
- Transactions are linked to specific Accounts.
- Cards are issued for specific Accounts and Customers.
- ATMs are associated with Branches.

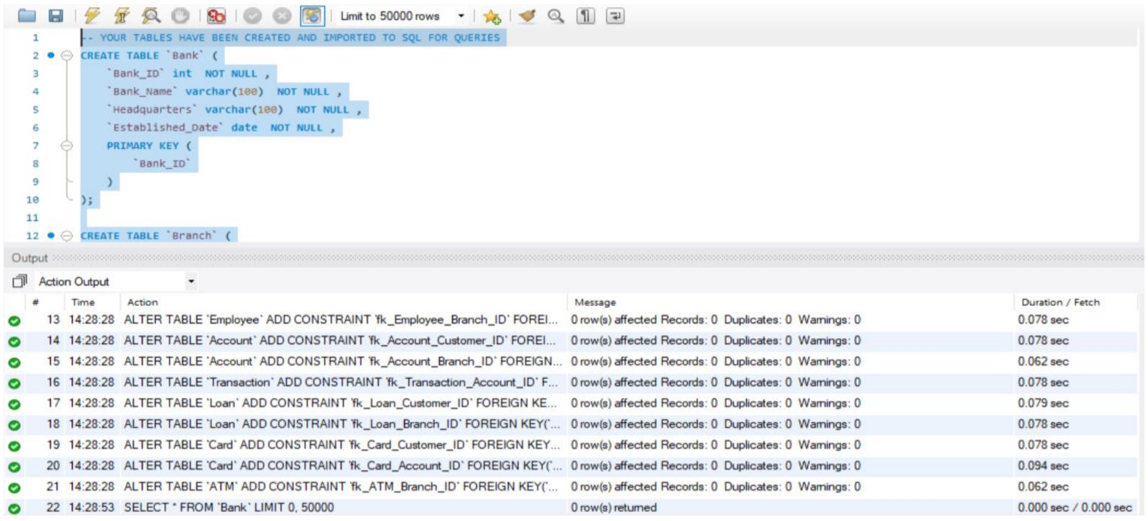


```
Name varchar(100)
Address varchar(150)
Phone varchar(20)
Email varchar(100)
Date_of_Birth date

Account
-
Account_ID int PK
Customer_ID int FK >- Customer.Customer_ID
Branch_ID int FK >- Branch.Branch_ID
Account_Type varchar(50)
Balance decimal(15,2)
Date_Opened date

Transaction
-
Transaction_ID int PK
Account_ID int FK >- Account.Account_ID
Transaction_Type varchar(50)
Amount decimal(15,2)
Date datetime
Description varchar(255)

Loan
-
Loan_ID int PK
Customer_ID int FK >- Customer.Customer_ID
Branch_ID int FK >- Branch.Branch_ID
Loan_Type varchar(50)
Amount decimal(15,2)
Interest_Rate decimal(5,2)
Start_Date date
End_Date date

Card
-
Card_ID int PK
Customer_ID int FK >- Customer.Customer_ID
Account_ID int FK >- Account.Account_ID
Card_Type varchar(50)
Issue_Date date
Expiry_Date date

ATM
-
ATM_ID int PK
```

```
No errors found. Good job!
```

7. Exporting to MySQL

Step 1: Download the SQL script exported from QuickDBD.
Step 2: Open MySQL Workbench or phpMyAdmin.
Step 3: Create a new database for the Bank Management System.
Step 4: Import the SQL script into the database.
Step 5: Run sample queries to test database functionality.



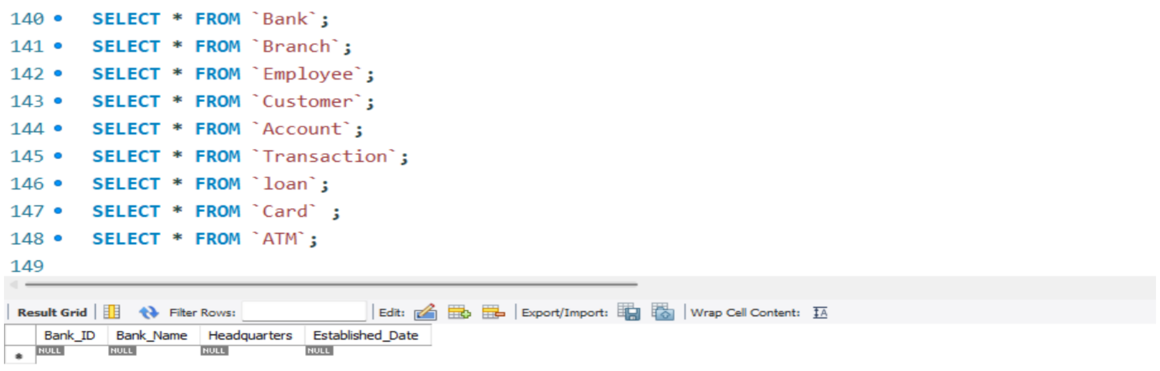
```sql
-- YOUR TABLES HAVE BEEN CREATED AND IMPORTED TO SQL FOR QUERIES
CREATE TABLE `Bank` (
    `Bank_ID` int  NOT NULL ,
    `Bank_Name` varchar(100)  NOT NULL ,
    `Headquarters` varchar(100)  NOT NULL ,
    `Established_Date` date  NOT NULL ,
    PRIMARY KEY (
        `Bank_ID`
    )
);

CREATE TABLE `Branch` (
```

```
   #  Time      Action                                                                 Message                                                        Duration / Fetch
  13  14:28:28  ALTER TABLE 'Employee' ADD CONSTRAINT 'fk_Employee_Branch_ID' FOREI...   0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0   0.078 sec
  14  14:28:28  ALTER TABLE 'Account' ADD CONSTRAINT 'fk_Account_Customer_ID' FOREI...   0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0   0.078 sec
  15  14:28:28  ALTER TABLE 'Account' ADD CONSTRAINT 'fk_Account_Branch_ID' FOREIGN...   0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0   0.062 sec
  16  14:28:28  ALTER TABLE 'Transaction' ADD CONSTRAINT 'fk_Transaction_Account_ID' F...  0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0   0.078 sec
  17  14:28:28  ALTER TABLE 'Loan' ADD CONSTRAINT 'fk_Loan_Customer_ID' FOREIGN KE...     0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0   0.079 sec
  18  14:28:28  ALTER TABLE 'Loan' ADD CONSTRAINT 'fk_Loan_Branch_ID' FOREIGN KEY('...    0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0   0.078 sec
  19  14:28:28  ALTER TABLE 'Card' ADD CONSTRAINT 'fk_Card_Customer_ID' FOREIGN KEY...    0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0   0.078 sec
  20  14:28:28  ALTER TABLE 'Card' ADD CONSTRAINT 'fk_Card_Account_ID' FOREIGN KEY('...   0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0   0.094 sec
  21  14:28:28  ALTER TABLE 'ATM' ADD CONSTRAINT 'fk_ATM_Branch_ID' FOREIGN KEY('...      0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0   0.062 sec
  22  14:28:53  SELECT * FROM 'Bank' LIMIT 0, 50000                                      0 row(s) returned                                              0.000 sec / 0.000 sec
```

8. Example SQL Queries

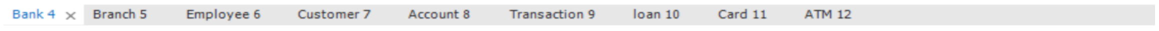
Example queries include:

- View all customers with their accounts.
- Retrieve all transactions for a specific account.
- Calculate total loans per branch.
- List all employees working in a particular branch.
- Show all active cards issued to a customer.



```sql
SELECT * FROM `Bank`;
SELECT * FROM `Branch`;
SELECT * FROM `Employee`;
SELECT * FROM `Customer`;
SELECT * FROM `Account`;
SELECT * FROM `Transaction`;
SELECT * FROM `loan`;
SELECT * FROM `Card` ;
SELECT * FROM `ATM`;
```

Bank_ID	Bank_Name	Headquarters	Established_Date
NULL	NULL	NULL	NULL



9. Conclusion

The Bank Management System (BMS) database, designed using QuickDBD and deployed on MySQL, delivers a robust, scalable, and secure foundation for modern banking operations. It integrates all critical components—banks, branches, employees, customers, accounts, transactions, loans, cards, and ATMs—while ensuring high standards of **data accuracy and referential integrity** through well-defined primary and foreign keys.

This design supports essential banking activities, including deposits, withdrawals, transfers, loan servicing, and card management, and is built to scale with growing transaction volumes and expanding digital services. The QuickDBD-to-MySQL workflow enabled efficient modeling and implementation, aligning with industry best practices for database lifecycle management.

Ultimately, this database not only meets current operational requirements but also provides a strong platform for **future growth, regulatory compliance, advanced analytics, and integration with emerging technologies** such as online banking and data visualization tools.

10. Future Enhancements and Data Analytics Integration

In the future, this Bank Management System database can be enhanced by implementing **advanced SQL queries and establishing additional relational mappings** to enable deeper analysis of banking operations. Complex queries such as customer behavior analytics, branch performance insights, and loan repayment trend analysis can be developed to support strategic decision-making.

Furthermore, integrating the database with **business intelligence tools like Microsoft Power BI** will enable dynamic data visualization, interactive dashboards, and real-time reporting. This will transform raw banking data into actionable insights, empowering management to monitor financial performance, detect anomalies, and forecast trends effectively.