BANK DATABASE MANAGEMENT WITH MY SQL

- The bank account management system is an application allowing customers to perform basic transactions from an automatic machine Bank, telephone, via a computer or with a Smartphone via the Internet. The system allows the customer to create an account, deposit / withdraw money from his account, as well as view reports from all accounts present.
  Customers can access the banks' website to view their account details and perform the transactions on their account according to their requirements. The connection is made by secure access.

- User

User ID
Username
Password
Name
Email
Phone

- Account

Account ID
Account Type
Balance

- Bank Database Management System Java Use

A bank management system implemented in Java typically involves using SQL databases to store and manage data related to customers, accounts, transactions, etc. Java provides JDBC (Java Database Connectivity) to interact with SQL databases. Here's a high-level overview of how you can implement a bank management system using Java and SQL

1. Bank Management System Class
    - This class serves as the main entry point for the bank management system.
    - It contains the JDBC connection parameters for connecting to the database.
    - Provides a method **getConnection()** to establish a connection to the database using JDBC.
    - Inside this class, nested classes for Customer, Account, and Transaction are defined.
2. Customer Class
    - Represents the Customer entity in the bank management system.
    - Defines attributes such as **customer_id**, **name**, **address**, **phone_number**, and **email**.
    - Contains a constructor to initialize a Customer object with the provided attributes.
    - It's intended to be expanded with additional methods for CRUD operations on customer data in the database.
3. Account Class
    - Represents the Account entity in the bank management system.
    - Defines attributes such as **account_number**, **customer_id**, **balance**, and **account_type**.
    - Contains a constructor to initialize an Account object with the provided attributes.
    - It's meant to be extended with methods for performing CRUD operations related to account data in the database.
4. Transaction Class
    - Represents the Transaction entity in the bank management system.
    - Defines attributes such as **transaction_id**, **account_number**, **transaction_type**, **amount**, and **transaction_date**.
    - Contains a constructor to initialize a Transaction object with the provided attributes.

- It's designed to be expanded with methods for managing transactions in the database, such as recording deposits, withdrawals, etc.

5. **Main Method**
   - Contains the **main()** method for testing the database connection.
   - Creates an instance of the BankManagementSystem class to establish a database connection and prints a success message if the connection is established successfully.

This structure provides a foundation for implementing a bank management system in Java, integrating it with a MySQL database using JDBC. You can further extend these classes with additional methods to handle more complex operations, such as querying customer account balances, transferring funds between accounts, etc.

➢ JDBC

A bank management system with SQL JDBC implementation involves designing and implementing a software system to manage various aspects of banking operations using SQL (Structured Query Language) for database management and JDBC (Java Database Connectivity) for Java-based communication with the database.

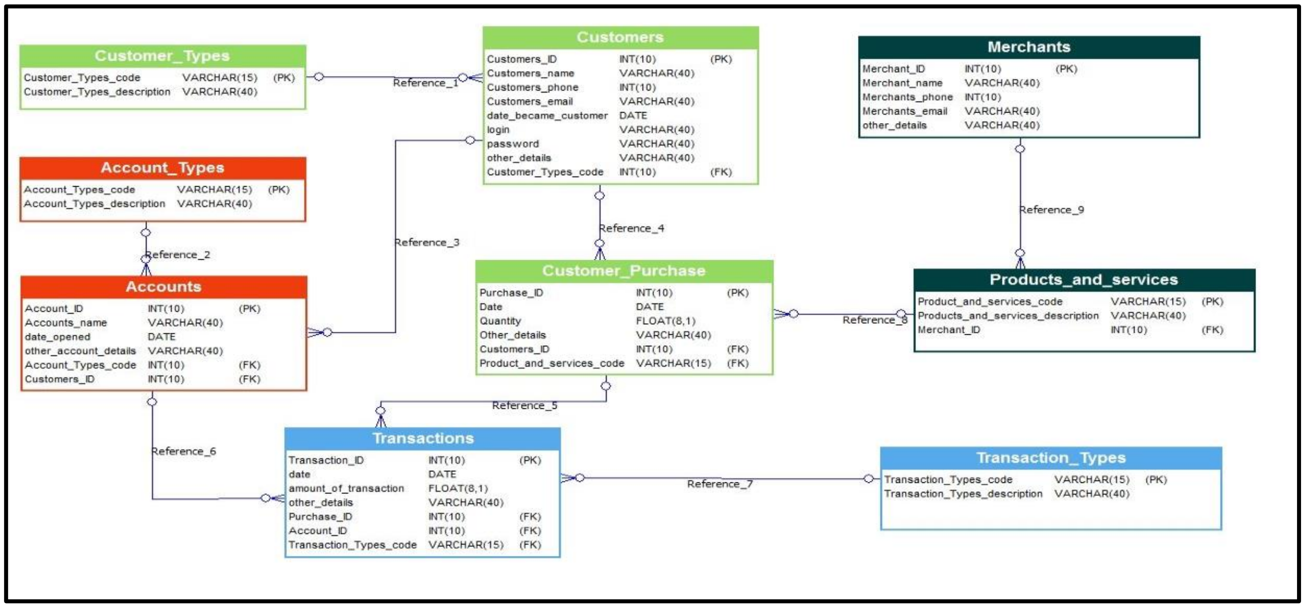
➢ Security

A bank management system with SQL security involves implementing various measures to protect sensitive data, ensure confidentiality, integrity, and availability, and prevent unauthorized access or malicious activities. Here's a breakdown of key security aspects.

➢ Error Handling

In a bank management system with SQL, error handling is crucial to ensure the reliability, integrity, and security of banking operations. Here's a definition and overview of error handling in such a system.

➢ Data Model Bank Management System



Customer_Types
Customer_Types_code VARCHAR(15) (PK)
Customer_Types_description VARCHAR(40)
Customers
Customers_ID INT(10) (PK)
Customers_name VARCHAR(40)
Customers_phone INT(10)
Customers_email VARCHAR(40)
date_became_customer DATE
login VARCHAR(40)
password VARCHAR(40)
other_details VARCHAR(40)
Customer_Types_code INT(10) (FK)
Merchants
Merchant_ID INT(10) (PK)
Merchant_name VARCHAR(40)
Merchants_phone INT(10)
Merchants_email VARCHAR(40)
other_details VARCHAR(40)
Account_Types
Account_Types_code VARCHAR(15) (PK)
Account_Types_description VARCHAR(40)
Accounts
Account_ID INT(10) (PK)
Accounts_name VARCHAR(40)
date_opened DATE
other_account_details VARCHAR(40)
Account_Types_code INT(10) (FK)
Customers_ID INT(10) (FK)
Customer_Purchase
Purchase_ID INT(10) (PK)
Date DATE
Quantity FLOAT(8,1)
Other_details VARCHAR(40)
Customers_ID INT(10) (FK)
Product_and_services_code VARCHAR(15) (FK)
Products_and_services
Product_and_services_code VARCHAR(15) (PK)
Products_and_services_description VARCHAR(40)
Merchant_ID INT(10) (FK)
Transactions
Transaction_ID INT(10) (PK)
date DATE
amount_of_transaction FLOAT(8,1)
other_details VARCHAR(40)
Purchase_ID INT(10) (FK)
Account_ID INT(10) (FK)
Transaction_Types_code VARCHAR(15) (FK)
Transaction_Types
Transaction_Types_code VARCHAR(15) (PK)
Transaction_Types_description VARCHAR(40)
Reference_1
Reference_2
Reference_3
Reference_4
Reference_5
Reference_6
Reference_7
Reference_8
Reference_9

➢ Table Structure

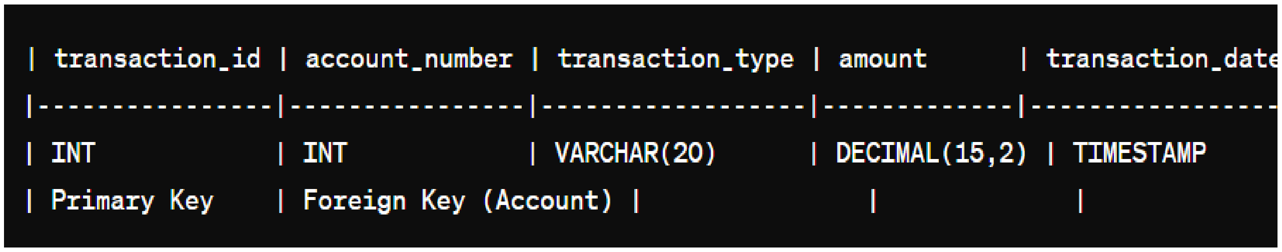
Customer Table

```
| customer_id | name        | address      | phone_number | email          |
|-------------|-------------|--------------|--------------|----------------|
| INT         | VARCHAR(100) | VARCHAR(255) | VARCHAR(15)  | VARCHAR(100)  |
| Primary Key |             |              |              |                |
```

Account Table

```
| account_number | customer_id | balance      | account_type |
|----------------|-------------|--------------|--------------|
| INT            | INT         | DECIMAL(15,2) | VARCHAR(50)  |
| Primary Key    | Foreign Key (Customer) |          |              |
```

Transaction Table



```
| transaction_id | account_number | transaction_type | amount       | transaction_date
|----------------|----------------|------------------|--------------|----------------
| INT            | INT            | VARCHAR(20)      | DECIMAL(15,2) | TIMESTAMP
| Primary Key    | Foreign Key (Account) |               |              |
```