***Banking Management Application Documentation***

***Overview***

The Banking Management Application is designed to manage various banking operations and functionalities, including account management, ATM management, loan management, and customer management. This application uses a MySQL database to store and manage data, and Java for the application logic and interactions with the database.

* Overview: [bank management system | Microsoft Whiteboard](https://app.whiteboard.microsoft.com/me/whiteboards/4fd2b390-fb8e-489b-af61-4fcdb5b7e05e)
* GitHub: [surendharS49/Bank-management-system (github.com)](https://github.com/surendharS49/Bank-management-system)

***Database Schema***

The database schema consists of the following tables:

**1. account**

* **Description**: Stores information about customer bank accounts.
* **Columns**:
  + accountID (INT, PK, AUTO\_INCREMENT): Unique identifier for the account.
  + customerID (INT, NULL): Foreign key linking to the customer.
  + accountType (VARCHAR(255), NULL): Type of the account (e.g., Savings, Checking).
  + balance (DECIMAL(15,2), NULL): Current balance of the account.
  + interestRate (DECIMAL(5,2), NULL): Interest rate applied to the account.
  + dateOpened (DATE, NULL): Date when the account was opened.
  + accountStatus (VARCHAR(255), NULL): Status of the account (e.g., Active, Closed).
* **Primary Key**: accountID
* **Foreign Key**: customerID references customer(customerId)

**2. atm**

* **Description**: Manages ATM information.
* **Columns**:
  + atmId (INT, PK, AUTO\_INCREMENT): Unique identifier for the ATM.
  + regionId (INT, NULL): Foreign key linking to the region.
  + balance (DECIMAL(15,2), NULL): Current balance in the ATM.
  + atmType (VARCHAR(50), NULL): Type of ATM (e.g., Deposit, Withdrawal).
  + status (VARCHAR(20), NULL): Status of the ATM (e.g., Operational, Out of Service).
  + lastServicedDate (DATE, NULL): Date when the ATM was last serviced.
  + capacity (INT, NULL): Capacity of the ATM.
* **Primary Key**: atmId
* **Foreign Key**: regionId references region(regionId)

**3. branch**

* **Description**: Contains information about bank branches.
* **Columns**:
  + branchId (INT, PK): Unique identifier for the branch.
  + name (VARCHAR(255), NOT NULL): Name of the branch.
  + address (VARCHAR(255), NOT NULL): Address of the branch.
  + phoneNumber (BIGINT, NOT NULL): Contact phone number for the branch.
  + managerId (INT, NULL): Foreign key linking to the branch manager.
  + regionId (INT, NULL): Foreign key linking to the region.
* **Primary Key**: branchId
* **Foreign Key**: regionId references region(regionId)

**4. card**

* **Description**: Manages card information.
* **Columns**:
  + cardId (INT, PK, AUTO\_INCREMENT): Unique identifier for the card.
  + cardNumber (VARCHAR(20), NOT NULL): Card number.
  + pin (VARCHAR(4), DEFAULT '1234'): PIN for the card.
  + cardholderName (VARCHAR(100), NOT NULL): Name of the cardholder.
  + expiryDate (DATE, NOT NULL): Expiry date of the card.
  + accountId (INT, NOT NULL): Foreign key linking to the account.
  + status (VARCHAR(20), NOT NULL): Status of the card (e.g., Active, Suspended).
  + cardIssuerId (INT, NULL): Foreign key linking to the card issuer.
* **Primary Key**: cardId
* **Foreign Key**: accountId references account(accountID), cardIssuerId references cardissuer(id)

**5. cardissuer**

* **Description**: Stores information about card issuers.
* **Columns**:
  + id (INT, PK, AUTO\_INCREMENT): Unique identifier for the card issuer.
  + cardIssuerName (VARCHAR(100), NOT NULL): Name of the card issuer.
  + cardType (VARCHAR(50), NOT NULL): Type of card issued (e.g., Credit, Debit).
* **Primary Key**: id

**6. customer**

* **Description**: Contains information about customers.
* **Columns**:
  + customerId (INT, PK, AUTO\_INCREMENT): Unique identifier for the customer.
  + name (VARCHAR(100), NOT NULL): Name of the customer.
  + address (VARCHAR(255), NULL): Address of the customer.
  + contactNumber (VARCHAR(15), NULL): Contact number of the customer.
  + email (VARCHAR(100), NULL): Email address of the customer.
  + dateOfBirth (DATE, NULL): Date of birth of the customer.
  + occupation (VARCHAR(50), NULL): Occupation of the customer.
  + accountStatus (VARCHAR(30), NOT NULL): Status of the customer’s account (e.g., Active, Inactive).
  + aadhaarNumber (VARCHAR(12), NULL): Aadhaar number of the customer.
* **Primary Key**: customerId

**7. loaninstallment**

* **Description**: Manages loan installments.
* **Columns**:
  + installmentId (INT, PK, AUTO\_INCREMENT): Unique identifier for the installment.
  + loanId (INT, NOT NULL): Foreign key linking to the loan.
  + paymentAmount (DOUBLE, NOT NULL): Amount of the installment payment.
  + paymentDate (DATE, NOT NULL): Date of the installment payment.
  + remainingBalance (DOUBLE, NOT NULL): Remaining balance after the installment payment.
* **Primary Key**: installmentId
* **Foreign Key**: loanId references loan(loanId)

**8. employee**

* **Description**: Contains information about bank employees.
* **Columns**:
  + employeeId (INT, PK, AUTO\_INCREMENT): Unique identifier for the employee.
  + name (VARCHAR(255), NOT NULL): Name of the employee.
  + branchId (INT, NOT NULL): Foreign key linking to the branch.
  + address (VARCHAR(255), NOT NULL): Address of the employee.
  + contactNo (BIGINT, NOT NULL): Contact number of the employee.
  + email (VARCHAR(255), NOT NULL): Email address of the employee.
  + position (VARCHAR(255), NOT NULL): Position of the employee.
  + department (VARCHAR(255), NOT NULL): Department of the employee.
  + salary (INT, NOT NULL): Salary of the employee.
  + dateOfHired (DATE, NOT NULL): Date when the employee was hired.
* **Primary Key**: employeeId
* **Foreign Key**: branchId references branch(branchId)

**9. interest**

* **Description**: Stores interest rate information for loans and accounts.
* **Columns**:
  + interest\_id (INT, PK, AUTO\_INCREMENT): Unique identifier for the interest record.
  + loan\_type (VARCHAR(50), NULL): Type of loan.
  + interest\_rate (DECIMAL(5,2), NULL): Interest rate.
  + effective\_date (DATE, NULL): Date when the interest rate became effective.
  + description (VARCHAR(255), NULL): Description of the interest rate.
  + compound\_frequency (VARCHAR(50), NULL): Frequency of interest compounding.
  + minimum\_balance (DECIMAL(10,2), NULL): Minimum balance required.
* **Primary Key**: interest\_id

**10. loan**

* **Description**: Contains information about loans.
* **Columns**:
  + loanId (INT, PK, AUTO\_INCREMENT): Unique identifier for the loan.
  + accountId (INT, NOT NULL): Foreign key linking to the account.
  + loanAmount (DECIMAL(15,2), NOT NULL): Amount of the loan.
  + interestRate (DECIMAL(5,2), NOT NULL): Interest rate applied to the loan.
  + loanTerm (INT, NOT NULL): Term of the loan in months.
  + loanStatus (VARCHAR(50), NOT NULL): Status of the loan (e.g., Approved, Rejected).
  + startDate (DATE, NOT NULL): Start date of the loan.
  + endDate (DATE, NOT NULL): End date of the loan.
  + loanType (VARCHAR(50), NOT NULL): Type of loan.
  + outstandingBalance (DECIMAL(15,2), NOT NULL): Outstanding balance of the loan.
  + paymentSchedule (VARCHAR(50), NULL): Payment schedule.
  + lateFee (DECIMAL(10,2), NULL): Late fee applicable.
  + purpose (VARCHAR(255), NULL): Purpose of the loan.
* **Primary Key**: loanId
* **Foreign Key**: accountId references account(accountID)

**11. region**

* **Description**: Manages regional information for the bank’s operations.
* **Columns**:
  + regionId (INT, PK): Unique identifier for the region.
  + name (VARCHAR(255), NOT NULL): Name of the region.
  + description (TEXT, NULL): Description of the region.
  + phoneNumber (BIGINT, NULL): Contact phone number for the region.
  + email (VARCHAR(255), NULL): Email address for the region.
* **Primary Key**: regionId

**12. transactions**

* **Description**: Records all transactions performed on accounts.
* **Columns**:
  + transactionID (INT, PK, AUTO\_INCREMENT): Unique identifier for the transaction.
  + accountID (INT, NOT NULL): Foreign key linking to the account involved in the transaction.
  + amount (DOUBLE, NOT NULL): Amount of the transaction.
  + transactionType (VARCHAR(10), NULL): Type of the transaction (e.g., Credit, Debit).
  + transactionDate (DATE, NOT NULL): Date of the transaction.
  + transactionDescription (VARCHAR(255), NULL): Description of the transaction.
  + loanID (INT, NULL): Foreign key linking to the loan (if applicable).
  + ReceiverAccountID (INT, NULL): Foreign key linking to the receiving account (if applicable).
* **Primary Key**: transactionID
* **Foreign Key**: accountID references account(accountID), loanID references loan(loanId)

**13. user**

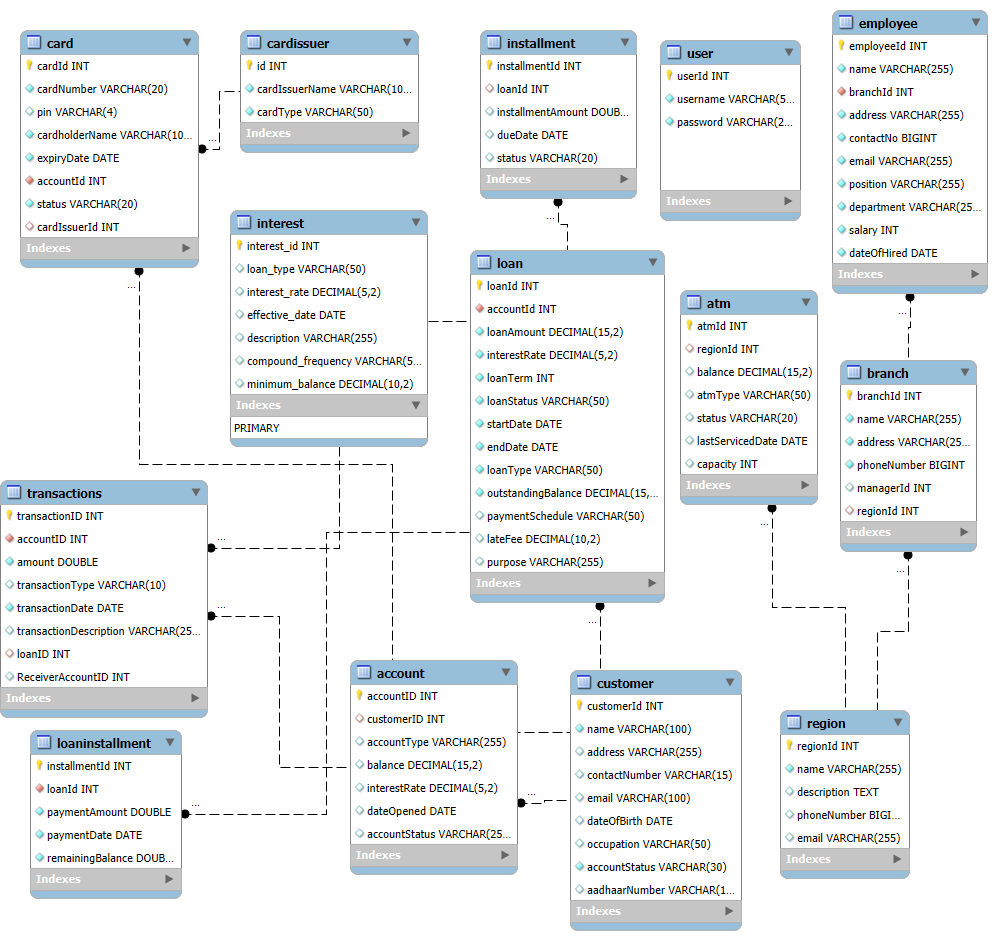
* **Description**: Manages user authentication and login credentials.
* **Columns**:
  + userId (INT, PK, AUTO\_INCREMENT): Unique identifier for the user.
  + username (VARCHAR(50), NOT NULL, UNIQUE): Username for login.
  + password (VARCHAR(255), NOT NULL): Encrypted password for the user.
* **Primary Key**: userId

***Application Features***

The Banking Management Application offers various features to handle banking operations efficiently:

* **Account Management**:
  + Create, update, and manage customer accounts.
  + Track account balances, account types, and status.
  + Handle account-related transactions and updates.
* **ATM Management**:
  + Monitor and manage ATMs across different regions.
  + Update ATM status, balance, and capacity.
  + Track servicing schedules and ATM types.
* **Branch Management**:
  + Manage branch details, including address and contact information.
  + Track branch-specific employees and operations.
  + Associate branches with specific regions.
* **Card Management**:
  + Issue and manage customer cards, including PINs and expiry dates.
  + Track card statuses and link cards to customer accounts.
* **Customer Management**:
  + Maintain customer profiles, including contact information and account status.
  + Manage customer-related transactions and services.
* **Loan Management**:
  + Process and manage loan applications.
  + Track loan details, such as amount, interest rate, and outstanding balance.
  + Handle loan installments and payment schedules.
* **Employee Management**:
  + Manage employee records, including personal details, salary, and position.
  + Track employee associations with branches and departments.
* **Interest Management**:
  + Update and manage interest rates for different loan types and accounts.
  + Track effective dates and compound frequencies.
* **Transaction Management**:
  + Record and track all financial transactions.
  + Manage transaction details such as amount, type, and description.
* **User Authentication**:
  + Secure user login and authentication using username and password.
  + Manage user credentials and access levels.

***Relationship Overview***

******

Here is a textual representation of how these tables are connected:

* **customer** → **account**: One-to-Many
  + A customer can have multiple accounts, but each account is associated with one customer.
* **account** → **card**: One-to-Many
  + An account can have multiple cards, but each card is associated with one account.
* **cardissuer** → **card**: One-to-Many
  + A card issuer can issue multiple cards, but each card has one issuer.
* **account** → **loan**: One-to-Many
  + An account can have multiple loans, but each loan is associated with one account.
* **loan** → **loaninstallment**: One-to-Many
  + A loan can have multiple installments, but each installment is associated with one loan.
* **branch** → **employee**: One-to-Many
  + A branch can have multiple employees, but each employee works at one branch.
* **region** → **branch**: One-to-Many
  + A region can have multiple branches, but each branch is located in one region.
* **region** → **atm**: One-to-Many
  + A region can have multiple ATMs, but each ATM is located in one region.
* **account** → **transactions**: One-to-Many
  + An account can have multiple transactions, but each transaction is associated with one account.
* **loan** → **transactions**: One-to-Many
  + A loan can have multiple transactions, but each transaction is associated with one loan.

***Application Architecture***

* **Database**: MySQL database to store and manage data related to accounts, transactions, loans, customers, and other banking operations.
* **Backend**: Java-based application logic to handle business processes, data interactions, and system operations.

***Installation and Setup***

1. **Database Setup**:
   * Install MySQL server.
   * Create a new database for the Banking Management Application.
   * Execute the provided SQL scripts to create the necessary tables and relationships.
2. **Java Application Setup**:
   * Ensure Java Development Kit (JDK) is installed.
   * Configure the application to connect to the MySQL database.
   * Compile and run the Java application to interact with the database.

***Conclusion***

The Banking Management Application is a comprehensive system designed to manage various aspects of banking operations. With robust features for account management, loan processing, and transaction handling, it provides an efficient solution for banking institutions to manage their operations effectively. The integration of Java and MySQL ensures a reliable and scalable application architecture.