Kotak Bank Database

Introduction

The **Kotak Bank Database System** is a robust and scalable MySQL-based database designed to manage banking operations efficiently. It includes tables for customers, accounts, branches, employees, transactions, and loans, enabling seamless data management and analytics for a banking environment.

Project Purpose

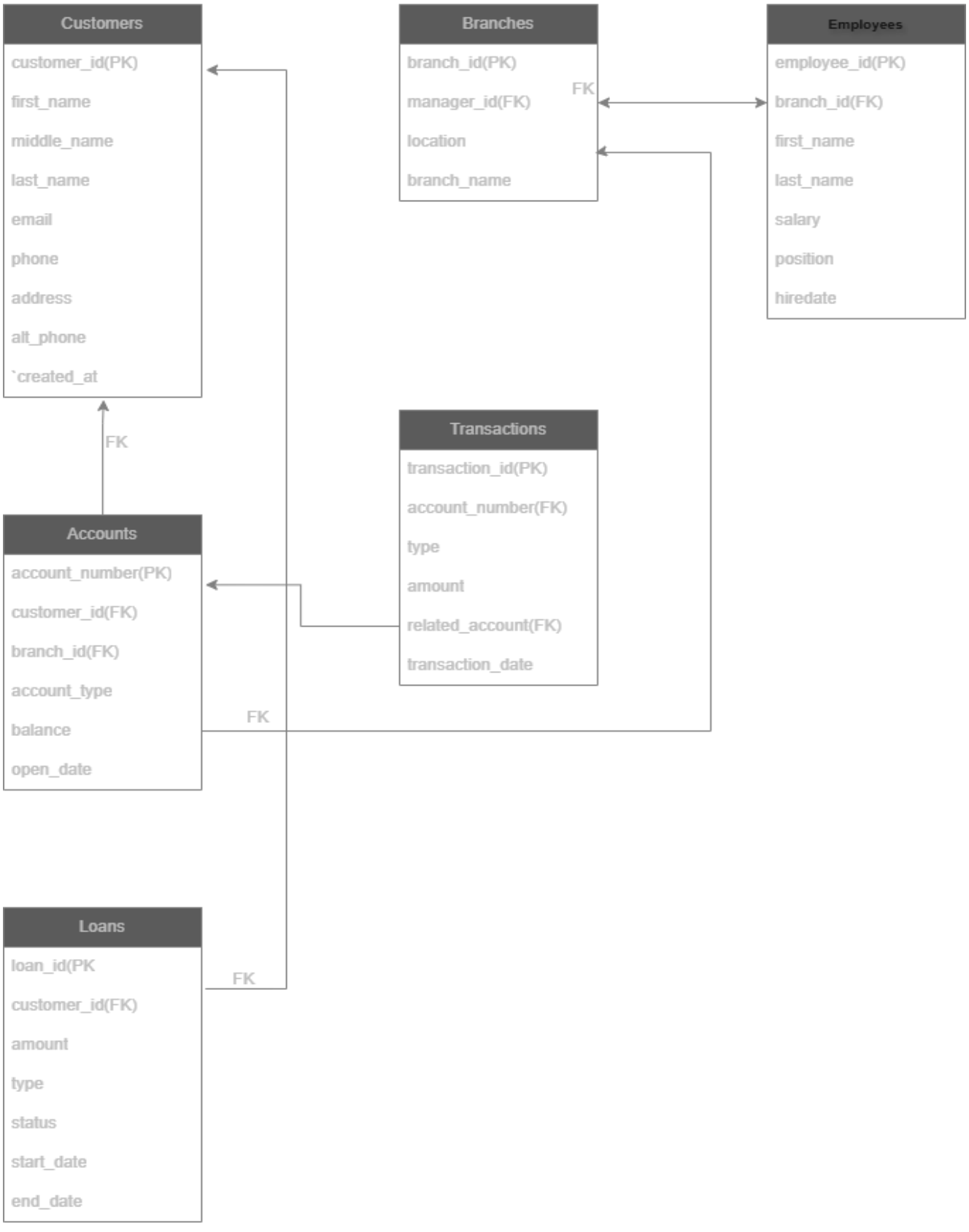
The purpose of this project is to:

1. **Centralize Banking Data**: Store and manage customer, account, and transaction data in a structured manner.
2. **Enable Analytics**: Provide sample SQL queries for insights into customer behavior, transaction trends, and loan approvals.
3. **Ensure Scalability**: Design a normalized database schema that can handle future expansions.

Features

1. **Comprehensive Schema**:
    - 6+ tables: Customers, Accounts, Branches, Employees, Transactions, and Loans.
    - Relationships between tables (e.g., customer-account, account-transaction).
2. **Data Validation**:
    - Constraints (e.g., UNIQUE email, NOT NULL phone).
    - Foreign keys for referential integrity.
3. **Sample Queries**:
    - Retrieve customer transaction history.
    - Calculate total deposits, withdrawals, and transfers.
    - Analyze loan approvals and rejections.
4. **Scalability**:
    - Modular design for adding new features (e.g., credit cards, investments).

ER-Diagram


Customers
customer_id(PK)
first_name
middle_name
last_name
email
phone
address
alt_phone
`created_at
Branches
branch_id(PK)
manager_id(FK)
location
branch_name
FK
Employees
employee_id(PK)
branch_id(FK)
first_name
last_name
salary
position
hiredate
FK
Transactions
transaction_id(PK)
account_number(FK)
type
amount
related_account(FK)
transaction_date
Accounts
account_number(PK)
customer_id(FK)
branch_id(FK)
account_type
balance
open_date
FK
FK
Loans
loan_id(PK
customer_id(FK)
amount
type
status
start_date
end_date
FK

Database Schema

1. Customers Table

```
CREATE TABLE Customers (
    customer_id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(50) NOT NULL,
    middle_name VARCHAR(50),
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE,
    phone VARCHAR(15) NOT NULL,
    alt_phone VARCHAR(15),
    address TEXT
);
```

2. Branches Table

```
CREATE TABLE Branches (
    branch_id INT PRIMARY KEY AUTO_INCREMENT,
    branch_name VARCHAR(100) NOT NULL,
    location VARCHAR(100) NOT NULL
);
```

3. Employees Table

```
CREATE TABLE Employees (
    employee_id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    branch_id INT,
    position VARCHAR(50),
    salary DECIMAL(10,2),
    hire_date DATE NOT NULL,
    FOREIGN KEY (branch_id) REFERENCES Branches(branch_id)
);
```

4. Accounts Table

```
CREATE TABLE Accounts (
   account_number VARCHAR(20) PRIMARY KEY,
   customer_id INT NOT NULL,
   account_type ENUM('Savings', 'Current', 'FD') NOT NULL,
   balance DECIMAL(15,2) DEFAULT 0.00,
   branch_id INT,
   opened_date DATE NOT NULL,
   FOREIGN KEY (customer_id) REFERENCES Customers(customer_id),
   FOREIGN KEY (branch_id) REFERENCES Branches(branch_id)
);
```

5. Transactions Table

```
CREATE TABLE Transactions (
   transaction_id INT PRIMARY KEY AUTO_INCREMENT,
   account_number VARCHAR(20) NOT NULL,
   type ENUM('Deposit', 'Withdrawal', 'Transfer') NOT NULL,
   amount DECIMAL(10,2) NOT NULL,
   transaction_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   related_account VARCHAR(20),
   FOREIGN KEY (account_number) REFERENCES Accounts(account_number)
);
```

6. Loans Table

```
CREATE TABLE Loans (
   loan_id INT PRIMARY KEY AUTO_INCREMENT,
   customer_id INT NOT NULL,
   amount DECIMAL(10,2) NOT NULL,
   type ENUM('Personal', 'Home', 'Car') NOT NULL,
   status ENUM('Pending', 'Approved', 'Rejected') DEFAULT 'Pending',
   start_date DATE NOT NULL,
   end_date DATE,
   FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);
```

Sample Queries

1. Retrieve Customer Transaction History

```
SELECT
  c.first_name, c.last_name, t.type, t.amount, t.transaction_date
FROM
  Customers c
JOIN
  Accounts a ON c.customer_id = a.customer_id
JOIN
  Transactions t ON a.account_number = t.account_number
WHERE
  c.customer_id = 1;
```

2. Calculate Total Deposits, Withdrawals, and Transfers

```
SELECT
  type, SUM(amount) AS total_amount
FROM
  Transactions
GROUP BY
  type;
```

3. Analyze Loan Approvals and Rejections

```
SELECT
  status, COUNT(*) AS total_loans
FROM
  Loans
GROUP BY
  status;
```

4. Find Customers with All Three Account Types

```
SELECT
   c.customer_id, c.first_name, c.last_name
FROM
   Customers c
JOIN
   Accounts a ON c.customer_id = a.customer_id
GROUP BY
   c.customer_id
HAVING
   COUNT(DISTINCT a.account_type) = 3;
```

Future Scope

1. **Integration with Python**: Use Python for advanced analytics and visualization.
2. **Dashboard Development**: Build a web-based dashboard for real-time insights.
3. **NoSQL Integration**: Add support for unstructured data (e.g., customer feedback).
4. **Automation**: Automate report generation using stored procedures.
5. **Security Enhancements**: Implement encryption for sensitive data.

How to Contribute

1. **Star the Repo**: Show your support by starring the repository.
2. **Fork the Project**: Create your own copy to experiment with.
3. **Submit PRs**: Contribute new features, bug fixes, or documentation improvements.
4. **Report Issues**: Help improve the project by reporting bugs or suggesting enhancements.

Conclusion

The **Kotak Bank Database System** is a comprehensive and scalable solution for managing banking operations. It provides a solid foundation for data management, analytics, and future expansions. Contributions and feedback are welcome to make this project even better!