Day 7: Banking & Financial Transactions

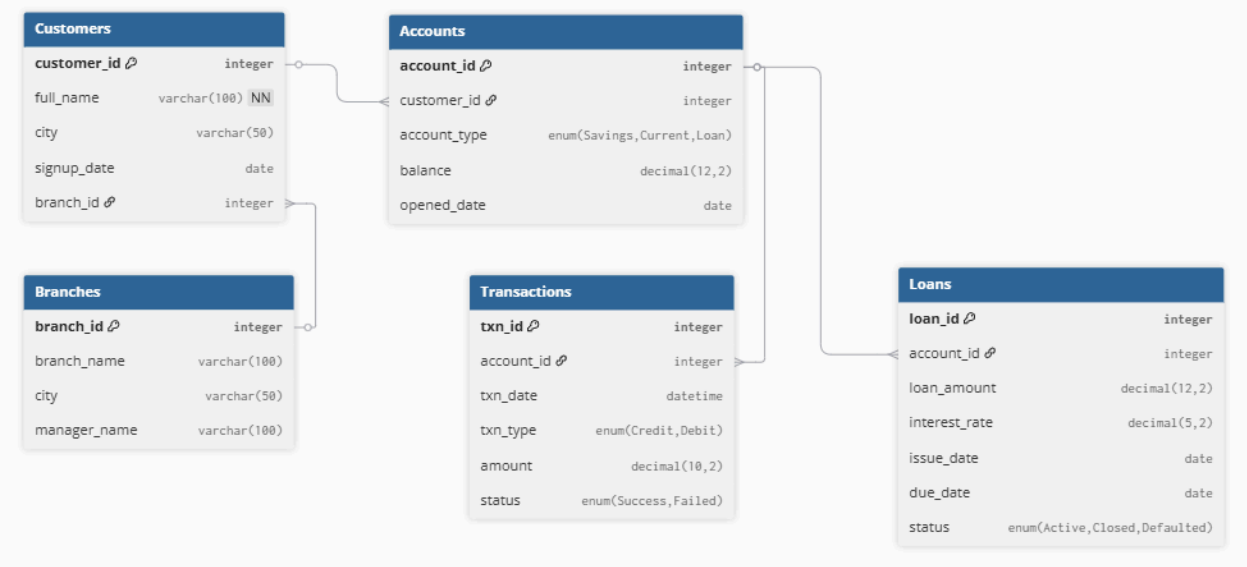
🎯 Business Context

A modern digital bank wants to analyze its **customer accounts, transactions, loans, and branches**.

Your task as a data engineer is to build queries that help:

- Detect **fraudulent transactions**
- Evaluate **customer activity and loan performance**
- Measure **branch profitability and transaction trends**

🗄️ Database Schema: FinBankDB



1️⃣ Branches

```
CREATE TABLE Branches (
    branch_id INT PRIMARY KEY,
    branch_name VARCHAR(100),
```

```
    city VARCHAR(50),
    manager_name VARCHAR(100)
);
```

2️⃣ Customers

```
CREATE TABLE Customers (
    customer_id INT PRIMARY KEY,
    full_name VARCHAR(100) NOT NULL,
    city VARCHAR(50),
    signup_date DATE,
    branch_id INT,
    FOREIGN KEY (branch_id) REFERENCES Branches(branch_id)
);
```

3️⃣ Accounts

```
CREATE TABLE Accounts (
    account_id INT PRIMARY KEY,
    customer_id INT,
    account_type ENUM('Savings', 'Current', 'Loan'),
    balance DECIMAL(12,2),
    opened_date DATE,
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);
```

4️⃣ Transactions

```
CREATE TABLE Transactions (
    txn_id INT PRIMARY KEY,
    account_id INT,
```

```
    txn_date DATETIME,
    txn_type ENUM('Credit', 'Debit'),
    amount DECIMAL(10,2),
    status ENUM('Success', 'Failed'),
    FOREIGN KEY (account_id) REFERENCES Accounts(account_id)
);
```

5️⃣ Loans

```
CREATE TABLE Loans (
    loan_id INT PRIMARY KEY,
    account_id INT,
    loan_amount DECIMAL(12,2),
    interest_rate DECIMAL(5,2),
    issue_date DATE,
    due_date DATE,
    status ENUM('Active', 'Closed', 'Defaulted'),
    FOREIGN KEY (account_id) REFERENCES Accounts(account_id)
);
```

📊 ERD Overview (Textual)

```
Branches (1)——< (M) Customers (1)——< (M) Accounts (1)——< (M) Transa
ctions
                                    |
                                    └——— Loans (1 per account)
```

🧮 Sample Data

```
INSERT INTO Branches VALUES
(1, 'Connaught Place', 'Delhi', 'Anil Mehta'),
(2, 'Andheri East', 'Mumbai', 'Pooja Iyer'),
(3, 'MG Road', 'Bangalore', 'Rahul Nair');

INSERT INTO Customers VALUES
(1, 'Rohit Sharma', 'Delhi', '2022-01-15', 1),
(2, 'Neha Verma', 'Mumbai', '2022-03-10', 2),
(3, 'Aman Gupta', 'Bangalore', '2023-01-01', 3),
(4, 'Sanya Kapoor', 'Delhi', '2023-04-20', 1),
(5, 'Raj Patel', 'Mumbai', '2023-02-10', 2);

INSERT INTO Accounts VALUES
(101, 1, 'Savings', 75000.00, '2022-01-16'),
(102, 1, 'Loan', 0.00, '2022-02-01'),
(103, 2, 'Savings', 50000.00, '2022-03-15'),
(104, 3, 'Current', 200000.00, '2023-01-05'),
(105, 4, 'Savings', 30000.00, '2023-04-25'),
(106, 5, 'Loan', 0.00, '2023-02-15');

INSERT INTO Transactions VALUES
(1, 101, '2024-09-01 10:00:00', 'Credit', 10000.00, 'Success'),
(2, 101, '2024-09-02 09:30:00', 'Debit', 2000.00, 'Success'),
(3, 103, '2024-09-02 11:00:00', 'Debit', 70000.00, 'Failed'),
(4, 104, '2024-09-03 14:30:00', 'Debit', 15000.00, 'Success'),
(5, 105, '2024-09-03 15:00:00', 'Credit', 12000.00, 'Success'),
(6, 101, '2024-09-03 15:30:00', 'Debit', 95000.00, 'Success'),
(7, 106, '2024-09-04 12:00:00', 'Credit', 200000.00, 'Success');

INSERT INTO Loans VALUES
(1, 102, 250000.00, 8.5, '2022-02-01', '2027-02-01', 'Active'),
(2, 106, 300000.00, 9.0, '2023-02-20', '2028-02-20', 'Active');
```

🧠 5 SQL Questions (Advanced)

🟢 (Easy)

Q1: List each branch and the number of customers registered in it.

Hint: Use `COUNT(customer_id)` grouped by branch.

🟡 (Medium)

Q2: Find all customers whose **failed transactions** exceed **2**.

Hint: Filter on `status='Failed'` and use `HAVING COUNT(txn_id) > 2`.

🟠 (Hard)

Q3: Identify **suspicious accounts** where a single debit transaction exceeds the **average debit amount** by more than **3×** for that branch.

Hint: Use subquery or CTE comparing each `amount` to average per branch.

🔵 (Difficult)

Q4: For each customer, calculate their **total transaction volume**, **net balance change** (credits - debits), and **most recent transaction date.**

Hint: Use `SUM(CASE WHEN txn_type='Credit' THEN amount ELSE 0 END)` and `MAX(txn_date)` grouped by customer.

🔴 (Expert)

Q5: Using a **window function**, rank customers within each branch based on their **total transaction volume** (credits + debits combined).

Display: `branch_name, customer_name, total_volume, rank_in_branch`.

Hint: Use `RANK() OVER (PARTITION BY branch_id ORDER BY SUM(amount) DESC)`.

⚙️ Advanced Optimization & Real-World Tips

- Create **indexes** on `(branch_id, customer_id)` and `(account_id, txn_date)` for efficient analytics.

- Use **partitioning by txn_date** for faster aggregation queries.
- Regularly **archive historical transactions** for performance.
- In fraud detection models, you can **materialize suspicious pattern tables** using periodic batch ETL jobs.