CSA-0593-DBMS

ASSIGNMENT-1

NAME : S.VARSHA

REG NO : 192311425

DATE : 14/11/2024

1. Multi-Branch Banking System with Loan and Interest Calculation

Design a database to support a multi-branch banking system, handling customer accounts, loans, and interest calculations.

Requirements:

Model tables for branches, accounts, customers, loans, and transactions.

Write stored procedures to handle loan approvals, calculate monthly interest, and process account statements.

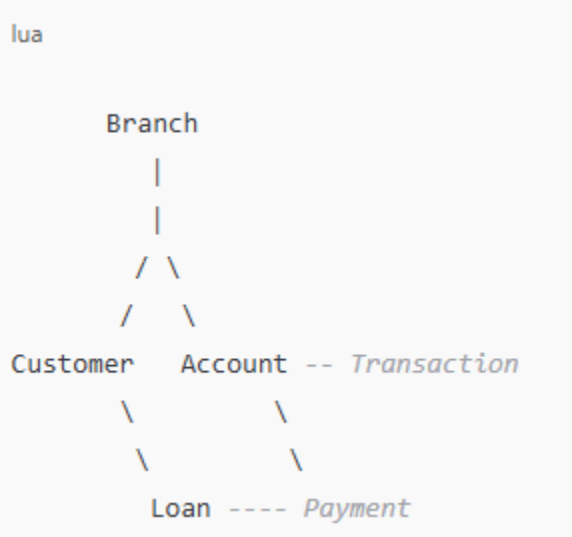
Implement triggers to update loan balances and notify customers of due payments.

Write SQL queries to analyze loan performance, interest income, and branch-wise profitability.

ANSWER :

Conceptual ER diagram

This conceptual ER model will form the basis for the database design and support the multi-branch banking system's operational and analytical requirements.



Database Schema Design

Branches Table

Stores information about different bank branches.

sql

Copy code

CREATE TABLE Branches (

BranchID INT PRIMARY KEY,

BranchName VARCHAR(100),

Location VARCHAR(100),

ManagerID INT

);

Customers Table

Stores customer data, including personal and contact information.

Column Name	Data Type	Description
branch_id	INT	Primary Key, Unique Branch Identifi
branch_name	VARCHAR(100)	Name of the Branch
branch_location	VARCHAR(200)	Branch Address or Location
phone_number	VARCHAR(15)	Branch Contact Number
manager_name	VARCHAR(100)	Branch Manager Name

sql

Copy code

```
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    FullName VARCHAR(100),
    Address VARCHAR(150),
    PhoneNumber VARCHAR(15),
    Email VARCHAR(100),
```

BranchID INT,

FOREIGN KEY (BranchID) REFERENCES Branches(BranchID)

);

Accounts Table

Stores customer accounts, each linked to a specific branch.

Column Name	Data Type	Description
customer_id	INT	Primary Key, Unique Customer Identifi
first_name	VARCHAR(100)	First Name of the Customer
last_name	VARCHAR(100)	Last Name of the Customer
email	VARCHAR(150)	Email Address
phone_number	VARCHAR(15)	Customer Contact Number
date_of_birth	DATE	Customer's Date of Birth
branch_id	INT	Foreign Key to Branches table

sql

Copy code

CREATE TABLE Accounts (

AccountID INT PRIMARY KEY,

AccountType VARCHAR(50),

Balance DECIMAL(15, 2),

CustomerID INT,

BranchID INT,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),

FOREIGN KEY (BranchID) REFERENCES Branches(BranchID)

);

Loans Table

Stores loan information, linked to a specific customer and branch.

Column Name	Data Type	Description
account_id	INT	Primary Key, Unique Account Identifier
account_type	VARCHAR(50)	Type of Account (e.g., Savings, Checkin
balance	DECIMAL(15,2)	Current Balance
created_date	DATE	Date of Account Creation
customer_id	INT	Foreign Key to Customers table
branch_id	INT	Foreign Key to Branches table

sql

Copy code

CREATE TABLE Loans (

```
    LoanID INT PRIMARY KEY,
    LoanType VARCHAR(50),
    Amount DECIMAL(15, 2),
    InterestRate DECIMAL(5, 2),
    MonthlyInstallment DECIMAL(15, 2),
    CustomerID INT,
    BranchID INT,
    Status VARCHAR(20),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
    FOREIGN KEY (BranchID) REFERENCES Branches(BranchID)
);
```

Transactions Table

Stores all transaction data for deposits, withdrawals, and loan payments.

Column Name	Data Type	Description
loan_id	INT	Primary Key, Unique Loan Identifie
loan_amount	DECIMAL(15, 2)	Total Loan Amount
interest_rate	DECIMAL(5, 2)	Interest Rate (Annual %)
loan_start_date	DATE	Date when the Loan was Issued
loan_end_date	DATE	Loan Maturity Date
balance	DECIMAL(15, 2)	Remaining Loan Balance
customer_id	INT	Foreign Key to Customers table
branch_id	INT	Foreign Key to Branches table

sql

Copy code

```
CREATE TABLE Transactions (
    TransactionID INT PRIMARY KEY,
    AccountID INT,
    LoanID INT,
    TransactionType VARCHAR(50), -- e.g., 'Deposit',
'Withdrawal', 'Loan Payment'
    Amount DECIMAL(15, 2),
    Date DATE,
```

FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID),

FOREIGN KEY (LoanID) REFERENCES Loans(LoanID)

);

Stored Procedures

Loan Approval Procedure

This procedure will approve or reject a loan based on criteria (e.g., credit score, branch capacity).

Column Name	Data Type	Description
transaction_id	INT	Primary Key, Unique Transaction ID
transaction_type	VARCHAR(50)	Type of Transaction (Deposit, Withdra
amount	DECIMAL(15, 2)	Transaction Amount
transaction_date	DATE	Date and Time of Transaction
account_id	INT	Foreign Key to Accounts table

sql

Copy code

CREATE PROCEDURE ApproveLoan(IN loan_id INT, IN approval_status VARCHAR(20))

BEGIN

UPDATE Loans

```
    SET Status = approval_status
    WHERE LoanID = loan_id;
END;
```

Monthly Interest Calculation

Calculates and updates monthly interest on each loan.

sql

Copy code

```
CREATE PROCEDURE CalculateMonthlyInterest()
BEGIN
    DECLARE monthly_interest DECIMAL(15, 2);
    DECLARE finished INT DEFAULT 0;
    DECLARE loan_cursor CURSOR FOR SELECT LoanID, Amount, InterestRate FROM Loans;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;

    OPEN loan_cursor;
    FETCH loan_cursor INTO loan_id, amount, interest_rate;

    WHILE finished = 0 DO
        SET monthly_interest = amount * (interest_rate / 100) / 12;
```

```
        UPDATE Loans
        SET MonthlyInstallment = monthly_interest
        WHERE LoanID = loan_id;

        FETCH loan_cursor INTO loan_id, amount, interest_rate;
    END WHILE;
    CLOSE loan_cursor;
END;
```

Process Account Statement

Generates account statements for a customer by listing all transactions within a specified period.

sql

Copy code

```
CREATE PROCEDURE GetAccountStatement(IN account_id INT, IN start_date DATE, IN end_date DATE)
BEGIN
    SELECT * FROM Transactions
    WHERE AccountID = account_id
    AND Date BETWEEN start_date AND end_date;
END;
```

Triggers

Update Loan Balances After Payment

This trigger updates loan balances and checks if the loan is fully paid.

sql

Copy code

```
CREATE TRIGGER AfterLoanPayment
AFTER INSERT ON Transactions
FOR EACH ROW
BEGIN
    IF NEW.TransactionType = 'Loan Payment' THEN
        UPDATE Loans
        SET Amount = Amount - NEW.Amount
        WHERE LoanID = NEW.LoanID;

        IF (SELECT Amount FROM Loans WHERE LoanID = NEW.LoanID) <= 0 THEN
            UPDATE Loans SET Status = 'Closed' WHERE LoanID = NEW.LoanID;
        END IF;
    END IF;
END;
```

Notify Customers of Due Payments

This trigger can be enhanced to send notifications for due payments.

sql

Copy code

```
CREATE TRIGGER NotifyDuePayments
BEFORE UPDATE ON Loans
FOR EACH ROW
BEGIN
    IF NEW.Status = 'Due' THEN
        -- Code for sending notification (pseudo code here)
        -- CALL SendNotification(NEW.CustomerID, 'Loan Payment Due');
    END IF;
END;
```

Analytical SQL Queries

Loan Performance Analysis

Retrieves all loans, their status, and remaining balance.

sql

Copy code

SELECT LoanID, LoanType, Status, Amount AS RemainingBalance

FROM Loans;

Interest Income Report

Calculates the total interest income for the bank on a monthly or annual basis.

sql

Copy code

SELECT SUM(MonthlyInstallment) AS TotalMonthlyInterest

FROM Loans;

Branch-Wise Profitability

Analyzes each branch's profitability by comparing total deposits, withdrawals, and interest income.

sql

Copy code

SELECT BranchID,

    SUM(CASE WHEN TransactionType = 'Deposit' THEN Amount ELSE 0 END) AS TotalDeposits,

    SUM(CASE WHEN TransactionType = 'Withdrawal' THEN Amount ELSE 0 END) AS TotalWithdrawals,

```
    SUM(MonthlyInstallment) AS TotalInterestIncome
FROM Transactions
JOIN Loans ON Transactions.LoanID = Loans.LoanID
GROUP BY BranchID;
```

Conclusion

This database design and implementation include:

Structured data storage for customers, accounts, loans, and branches.

Procedures for managing loans, calculating interest, and generating account statements.

Triggers to maintain data consistency and support customer notifications for due payments.

Analytical queries for financial reporting, providing insights into loan performance, interest income, and branch profitability.

This solution ensures efficient management of banking operations and allows the bank to maintain a centralized and comprehensive view of its financial standing and branch-wise profitability.