**Week-2**

**PL/SQL**

Exercise:Stored Procedures

-- Drop existing tables (optional for re-run)

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Transactions CASCADE CONSTRAINTS';

EXECUTE IMMEDIATE 'DROP TABLE Accounts CASCADE CONSTRAINTS';

EXECUTE IMMEDIATE 'DROP TABLE Loans CASCADE CONSTRAINTS';

EXECUTE IMMEDIATE 'DROP TABLE Customers CASCADE CONSTRAINTS';

EXECUTE IMMEDIATE 'DROP TABLE Employees CASCADE CONSTRAINTS';

EXCEPTION

WHEN OTHERS THEN NULL;

END;

/

-- Create Tables

CREATE TABLE Customers (

CustomerID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

DOB DATE,

Balance NUMBER,

LastModified DATE

);

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerID NUMBER,

AccountType VARCHAR2(20),

Balance NUMBER,

LastModified DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Transactions (

TransactionID NUMBER PRIMARY KEY,

AccountID NUMBER,

TransactionDate DATE,

Amount NUMBER,

TransactionType VARCHAR2(10),

FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)

);

CREATE TABLE Loans (

LoanID NUMBER PRIMARY KEY,

CustomerID NUMBER,

LoanAmount NUMBER,

InterestRate NUMBER,

StartDate DATE,

EndDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Employees (

EmployeeID NUMBER PRIMARY KEY,

Name VARCHAR2(100),

Position VARCHAR2(50),

Salary NUMBER,

Department VARCHAR2(50),

HireDate DATE

);

-- Create sequence for transactions

CREATE SEQUENCE Transactions\_seq START WITH 3 INCREMENT BY 1;

-- Insert sample data

INSERT INTO Customers VALUES (1, 'John Doe', TO\_DATE('1985-05-15', 'YYYY-MM-DD'), 1000, SYSDATE);

INSERT INTO Customers VALUES (2, 'Jane Smith', TO\_DATE('1990-07-20', 'YYYY-MM-DD'), 1500, SYSDATE);

INSERT INTO Accounts VALUES (1, 1, 'Savings', 1000, SYSDATE);

INSERT INTO Accounts VALUES (2, 2, 'Checking', 1500, SYSDATE);

INSERT INTO Transactions VALUES (1, 1, SYSDATE, 200, 'Deposit');

INSERT INTO Transactions VALUES (2, 2, SYSDATE, 300, 'Withdrawal');

INSERT INTO Loans VALUES (1, 1, 5000, 5, SYSDATE, ADD\_MONTHS(SYSDATE, 60));

INSERT INTO Employees VALUES (1, 'Alice Johnson', 'Manager', 70000, 'HR', TO\_DATE('2015-06-15', 'YYYY-MM-DD'));

INSERT INTO Employees VALUES (2, 'Bob Brown', 'Developer', 60000, 'IT', TO\_DATE('2017-03-20', 'YYYY-MM-DD'));

COMMIT;

-- Enable output

SET SERVEROUTPUT ON;

--------------------------------------------------------------------------------

-- Procedure 1: Apply 1% interest to all Savings accounts

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS

BEGIN

FOR acc IN (SELECT AccountID, Balance FROM Accounts WHERE AccountType = 'Savings') LOOP

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01),

LastModified = SYSDATE

WHERE AccountID = acc.AccountID;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('1% interest applied to savings accounts.');

END;

/

BEGIN

ProcessMonthlyInterest;

END;

/

--------------------------------------------------------------------------------

-- Procedure 2: Add bonus to all employees in a given department

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_dept IN VARCHAR2,

p\_bonus\_pct IN NUMBER

) AS

BEGIN

FOR emp IN (SELECT EmployeeID, Salary FROM Employees WHERE Department = p\_dept) LOOP

UPDATE Employees

SET Salary = Salary + (Salary \* p\_bonus\_pct / 100)

WHERE EmployeeID = emp.EmployeeID;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Bonus of ' || p\_bonus\_pct || '% applied to ' || p\_dept || ' department.');

END;

/

BEGIN

UpdateEmployeeBonus('IT', 10);

END;

/

--------------------------------------------------------------------------------

-- Procedure 3: Transfer funds between accounts

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_from\_acc IN NUMBER,

p\_to\_acc IN NUMBER,

p\_amount IN NUMBER

) AS

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance FROM Accounts WHERE AccountID = p\_from\_acc;

IF v\_balance >= p\_amount THEN

-- Deduct from sender

UPDATE Accounts

SET Balance = Balance - p\_amount,

LastModified = SYSDATE

WHERE AccountID = p\_from\_acc;

-- Add to receiver

UPDATE Accounts

SET Balance = Balance + p\_amount,

LastModified = SYSDATE

WHERE AccountID = p\_to\_acc;

-- Insert into Transactions

INSERT INTO Transactions VALUES (

Transactions\_seq.NEXTVAL, p\_from\_acc, SYSDATE, p\_amount, 'Transfer-Out'

);

INSERT INTO Transactions VALUES (

Transactions\_seq.NEXTVAL, p\_to\_acc, SYSDATE, p\_amount, 'Transfer-In'

);

DBMS\_OUTPUT.PUT\_LINE('Rs' || p\_amount || ' transferred from Account ' || p\_from\_acc || ' to Account ' || p\_to\_acc);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Insufficient funds in Account ' || p\_from\_acc);

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Account not found.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END;

/

BEGIN

TransferFunds(2, 1, 500);

END;

/

--------------------------------------------------------------------------------

-- View final data

SELECT \* FROM Accounts;

SELECT \* FROM Employees;

SELECT \* FROM Transactions;

Output:

A screenshot of a computer

Description automatically generated

OUTPUT:

