Week-2

PL SQL programming

**Exercise 5: Triggers**

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

-- CLEANUP OLD TABLES

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

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE AuditLog';

EXCEPTION

WHEN OTHERS THEN IF SQLCODE != -942 THEN RAISE; END IF;

END;

/

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Transactions';

EXCEPTION

WHEN OTHERS THEN IF SQLCODE != -942 THEN RAISE; END IF;

END;

/

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Accounts';

EXCEPTION

WHEN OTHERS THEN IF SQLCODE != -942 THEN RAISE; END IF;

END;

/

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Customers';

EXCEPTION

WHEN OTHERS THEN IF SQLCODE != -942 THEN RAISE; END IF;

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 AuditLog (

LogID NUMBER GENERATED ALWAYS AS IDENTITY,

TransactionID NUMBER,

LogDate DATE DEFAULT SYSDATE,

Message VARCHAR2(255)

);

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

-- 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'), 2000, SYSDATE);

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

INSERT INTO Accounts VALUES (2, 2, 'Checking', 2000, SYSDATE);

COMMIT;

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

-- Scenario 1: Automatically update LastModified on Customers

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

CREATE OR REPLACE TRIGGER UpdateCustomerLastModified

BEFORE UPDATE ON Customers

FOR EACH ROW

BEGIN

:NEW.LastModified := SYSDATE;

END;

/

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

-- Scenario 2: Maintain an audit log for all transactions

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

CREATE OR REPLACE TRIGGER LogTransaction

AFTER INSERT ON Transactions

FOR EACH ROW

BEGIN

INSERT INTO AuditLog (TransactionID, Message)

VALUES (:NEW.TransactionID, 'Transaction inserted for Account '||:NEW.AccountID);

END;

/

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

-- Scenario 3: Enforce business rules on deposits and withdrawals

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

CREATE OR REPLACE TRIGGER CheckTransactionRules

BEFORE INSERT ON Transactions

FOR EACH ROW

DECLARE

acc\_balance NUMBER;

BEGIN

SELECT Balance INTO acc\_balance FROM Accounts WHERE AccountID = :NEW.AccountID;

IF :NEW.TransactionType = 'Withdrawal' AND :NEW.Amount > acc\_balance THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient balance for withdrawal.');

ELSIF :NEW.TransactionType = 'Deposit' AND :NEW.Amount <= 0 THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Deposit amount must be positive.');

END IF;

END;

/

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

-- TEST THE TRIGGERS

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

SET SERVEROUTPUT ON

BEGIN

DBMS\_OUTPUT.PUT\_LINE('>> Update Customer to see LastModified change');

UPDATE Customers SET Balance = Balance + 500 WHERE CustomerID = 1;

COMMIT;

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('>> Insert valid Deposit transaction (should work)');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (1, 1, SYSDATE, 100, 'Deposit');

COMMIT;

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('>> Insert valid Withdrawal transaction (should work)');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (2, 1, SYSDATE, 200, 'Withdrawal');

COMMIT;

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('>> Try Withdrawal with insufficient funds (should fail)');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (3, 1, SYSDATE, 10000, 'Withdrawal');

EXCEPTION

WHEN OTHERS THEN

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

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('>> Try Deposit with negative amount (should fail)');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (4, 1, SYSDATE, -50, 'Deposit');

EXCEPTION

WHEN OTHERS THEN

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

END;

/

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

-- SEE THE AUDIT LOG

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

BEGIN

FOR rec IN (SELECT \* FROM AuditLog) LOOP

DBMS\_OUTPUT.PUT\_LINE('AuditLog: TransactionID='||rec.TransactionID||', Message='||rec.Message);

END LOOP;

END;

/

