**Exercise 2: Error Handling**

**Scenario 1:** Handle exceptions during fund transfers between accounts.

* + **Question:** Write a stored procedure **SafeTransferFunds** that transfers funds between two accounts. Ensure that if any error occurs (e.g., insufficient funds), an appropriate error message is logged and the transaction is rolled back.

**Scenario 2:** Manage errors when updating employee salaries.

* + **Question:** Write a stored procedure **UpdateSalary** that increases the salary of an employee by a given percentage. If the employee ID does not exist, handle the exception and log an error message.

**Scenario 3:** Ensure data integrity when adding a new customer.

* + **Question:** Write a stored procedure **AddNewCustomer** that inserts a new customer into the Customers table. If a customer with the same ID already exists, handle the exception by logging an error and preventing the insertion.

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

Name VARCHAR(100),

DOB DATE,

Balance DECIMAL(10, 2),

LastModified TIMESTAMP

);

CREATE TABLE Accounts (

AccountID INT PRIMARY KEY,

CustomerID INT,

AccountType VARCHAR(20),

Balance DECIMAL(10, 2),

LastModified TIMESTAMP,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Transactions (

TransactionID INT PRIMARY KEY,

AccountID INT,

TransactionDate TIMESTAMP,

Amount DECIMAL(10, 2),

TransactionType VARCHAR(10),

FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)

);

CREATE TABLE Loans (

LoanID INT PRIMARY KEY,

CustomerID INT,

LoanAmount DECIMAL(10, 2),

InterestRate DECIMAL(5, 2),

StartDate DATE,

EndDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

Name VARCHAR(100),

Position VARCHAR(50),

Salary DECIMAL(10, 2),

Department VARCHAR(50),

HireDate DATE

);

-- Insert into Customers

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES (1, 'Michael Scott', '1964-03-15', 2000, NOW());

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES (2, 'Pam Beesly', '1979-03-25', 3000, NOW());

-- Insert into Accounts

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (1, 1, 'Savings', 2000, NOW());

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES (2, 2, 'Checking', 3000, NOW());

-- Insert into Transactions

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

VALUES (1, 1, NOW(), 500, 'Deposit');

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

VALUES (2, 2, NOW(), 700, 'Withdrawal');

-- Insert into Loans

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES (1, 1, 10000, 4.50, NOW(), DATE\_ADD(NOW(), INTERVAL 7 YEAR));

-- Insert into Employees

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (1, 'Dwight Schrute', 'Assistant Manager', 80000, 'Sales', '2005-04-01');

INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES (2, 'Jim Halpert', 'Sales Representative', 75000, 'Sales', '2006-07-10');

DELIMITER //

DELIMITER //

CREATE PROCEDURE SecureFundTransfer(

IN sourceAccountID INT,

IN destinationAccountID INT,

IN amountToTransfer DECIMAL(10, 2)

)

BEGIN

DECLARE noFundsAvailable BOOL DEFAULT FALSE;

DECLARE accountMissing BOOL DEFAULT FALSE;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

-- Rollback transaction and log the error

ROLLBACK;

SELECT 'Transfer failed. Transaction has been rolled back.' AS ErrorMessage;

END;

START TRANSACTION;

-- Check if both accounts exist

IF (SELECT COUNT(\*) FROM Accounts WHERE AccountID = sourceAccountID) = 0 THEN

SET accountMissing = TRUE;

END IF;

IF (SELECT COUNT(\*) FROM Accounts WHERE AccountID = destinationAccountID) = 0 THEN

SET accountMissing = TRUE;

END IF;

IF accountMissing THEN

ROLLBACK;

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'One or both accounts do not exist';

END IF;

-- Check if the sourceAccount has sufficient balance

IF (SELECT Balance FROM Accounts WHERE AccountID = sourceAccountID) < amountToTransfer THEN

SET noFundsAvailable = TRUE;

END IF;

IF noFundsAvailable THEN

ROLLBACK;

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Insufficient funds in the source account';

END IF;

-- Execute the transfer

UPDATE Accounts SET Balance = Balance - amountToTransfer WHERE AccountID = sourceAccountID;

UPDATE Accounts SET Balance = Balance + amountToTransfer WHERE AccountID = destinationAccountID;

COMMIT;

SELECT 'Funds transferred successfully' AS SuccessMessage;

END //

DELIMITER ;

DELIMITER //

CREATE PROCEDURE ModifyEmployeeSalary(

IN employeeID INT,

IN increaseBy DECIMAL(10, 2)

)

BEGIN

DECLARE employeeAbsent BOOL DEFAULT FALSE;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

-- Log error and rollback

ROLLBACK;

SELECT 'Error occurred while updating the salary. Transaction rolled back.' AS ErrorMessage;

END;

START TRANSACTION;

-- Verify if employee exists

IF (SELECT COUNT(\*) FROM Employees WHERE EmployeeID = employeeID) = 0 THEN

SET employeeAbsent = TRUE;

END IF;

IF employeeAbsent THEN

ROLLBACK;

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Employee ID not found';

END IF;

-- Update the employee's salary

UPDATE Employees SET Salary = Salary + increaseBy WHERE EmployeeID = employeeID;

COMMIT;

SELECT 'Salary has been successfully updated' AS SuccessMessage;

END //

DELIMITER ;

DELIMITER //

CREATE PROCEDURE InsertNewCustomer(

IN customerID INT,

IN customerName VARCHAR(100),

IN birthDate DATE,

IN initialBalance DECIMAL(10, 2)

)

BEGIN

DECLARE customerExists BOOL DEFAULT FALSE;

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

-- Log error and rollback

ROLLBACK;

SELECT 'Failed to add new customer. Transaction rolled back.' AS ErrorMessage;

END;

START TRANSACTION;

-- Check if customer already exists

IF (SELECT COUNT(\*) FROM Customers WHERE CustomerID = customerID) > 0 THEN

SET customerExists = TRUE;

END IF;

IF customerExists THEN

ROLLBACK;

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Customer with the provided ID already exists';

END IF;

-- Insert the new customer

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES (customerID, customerName, birthDate, initialBalance, CURRENT\_TIMESTAMP);

COMMIT;

SELECT 'Customer added successfully' AS SuccessMessage;

END //

DELIMITER ;

DELIMITER //

SELECT COUNT(\*) FROM Employees WHERE EmployeeID = 1;

UPDATE Employees SET Salary = Salary + 1000 WHERE EmployeeID = 1;

SELECT \* FROM Employees WHERE EmployeeID = 1;

SELECT COUNT(\*) FROM Customers WHERE CustomerID = 3;

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES (3, 'New Customer', '2000-01-01', 5000, CURRENT\_TIMESTAMP);

SELECT \* FROM Customers WHERE CustomerID = 3;

CALL ModifyEmployeeSalary(1, 1000);

CALL InsertNewCustomer(3, 'New Customer', '2000-01-01', 5000);

SELECT \* FROM Employees WHERE EmployeeID = 1;

SELECT \* FROM Customers WHERE CustomerID = 3;

CALL ModifyEmployeeSalary(999, 1000.00);

CALL SecureFundTransfer(1, 2, 100.00);

SELECT \* FROM Accounts;

CALL SecureFundTransfer(1, 2, 5000.00);