**WEEK – 2**

**PLSQL**

**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.

**Scenario 1: SafeTransferFunds Procedure**

DECLARE

TYPE AccountRec IS RECORD (

acc\_id NUMBER,

balance NUMBER

);

TYPE AccountTable IS TABLE OF AccountRec INDEX BY PLS\_INTEGER;

accounts AccountTable;

PROCEDURE SafeTransferFunds(from\_acc NUMBER, to\_acc NUMBER, amount NUMBER) IS

from\_index PLS\_INTEGER := -1;

to\_index PLS\_INTEGER := -1;

BEGIN

-- Find indexes

FOR i IN accounts.FIRST .. accounts.LAST LOOP

IF accounts(i).acc\_id = from\_acc THEN

from\_index := i;

ELSIF accounts(i).acc\_id = to\_acc THEN

to\_index := i;

END IF;

END LOOP;

IF from\_index = -1 OR to\_index = -1 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'One or both accounts not found.');

END IF;

IF accounts(from\_index).balance < amount THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Insufficient funds.');

END IF;

-- Transfer

accounts(from\_index).balance := accounts(from\_index).balance - amount;

accounts(to\_index).balance := accounts(to\_index).balance + amount;

DBMS\_OUTPUT.PUT\_LINE('Transfer of ' || amount || ' from ' || from\_acc || ' to ' || to\_acc || ' successful.');

EXCEPTION

WHEN OTHERS THEN

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

END;

BEGIN

-- Sample accounts

accounts(1) := AccountRec(101, 5000);

accounts(2) := AccountRec(102, 3000);

-- Successful transfer

SafeTransferFunds(101, 102, 1000);

-- Failing transfer (insufficient funds)

SafeTransferFunds(102, 101, 10000);

END;

**Output:**

Transfer of 1000 from 101 to 102 successful.

Transfer Failed: ORA-20002: Insufficient funds.

**Scenario 2: Update Employee Salary with Error Handling**

DECLARE

TYPE EmployeeRec IS RECORD (

emp\_id NUMBER,

name VARCHAR2(50),

salary NUMBER

);

TYPE EmployeeTable IS TABLE OF EmployeeRec INDEX BY PLS\_INTEGER;

employees EmployeeTable;

PROCEDURE UpdateSalary(emp\_id NUMBER, percent NUMBER) IS

found BOOLEAN := FALSE;

BEGIN

FOR i IN employees.FIRST .. employees.LAST LOOP

IF employees(i).emp\_id = emp\_id THEN

employees(i).salary := employees(i).salary + (employees(i).salary \* percent / 100);

DBMS\_OUTPUT.PUT\_LINE('Salary updated for ' || employees(i).name ||

': New Salary = ' || employees(i).salary);

found := TRUE;

EXIT;

END IF;

END LOOP;

IF NOT found THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Employee ID not found.');

END IF;

EXCEPTION

WHEN OTHERS THEN

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

END;

BEGIN

-- Sample employees

employees(1) := EmployeeRec(1, 'Alice', 30000);

employees(2) := EmployeeRec(2, 'Bob', 40000);

-- Valid update

UpdateSalary(1, 10);

-- Invalid update

UpdateSalary(3, 5);

END;

**Output:**

Salary updated for Alice: New Salary = 33000

Update Failed: ORA-20003: Employee ID not found.

**Scenario 3: Add New Customer with Duplicate Check**

DECLARE

TYPE CustomerRec IS RECORD (

cust\_id NUMBER,

name VARCHAR2(100)

);

TYPE CustomerTable IS TABLE OF CustomerRec INDEX BY PLS\_INTEGER;

customers CustomerTable;

PROCEDURE AddNewCustomer(cust\_id NUMBER, name VARCHAR2) IS

duplicate BOOLEAN := FALSE;

BEGIN

FOR i IN customers.FIRST .. customers.LAST LOOP

IF customers(i).cust\_id = cust\_id THEN

duplicate := TRUE;

EXIT;

END IF;

END LOOP;

IF duplicate THEN

RAISE\_APPLICATION\_ERROR(-20004, 'Customer ID already exists.');

ELSE

customers(customers.LAST + 1) := CustomerRec(cust\_id, name);

DBMS\_OUTPUT.PUT\_LINE('Customer added: ' || name);

END IF;

EXCEPTION

WHEN OTHERS THEN

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

END;

BEGIN

-- Initial customers

customers(1) := CustomerRec(1001, 'John');

customers(2) := CustomerRec(1002, 'Mary');

-- Add new customer

AddNewCustomer(1003, 'David');

-- Try duplicate

AddNewCustomer(1001, 'Duplicate John');

END;

**Output:**

Customer added: David

Add Failed: ORA-20004: Customer ID already exists.