Database Programming with PL/SQL

7-3: Trapping User-Defined Exceptions Practice Activities

Vocabulary

Identify the vocabulary word for each definition below:

RAISE_APPLICATION_ERROR	A procedure used to return user-defined error messages from stored subprograms.
RAISE	Use this statement to raise a named exception.
User-Defined error	These errors are not automatically raised by the Oracle Server, but are defined by the programmer and are specific to the programmer's code.

Try It / Solve It

All the questions in this exercise use a copy of the employees table. Create this copy by running the following SQL statement:

```
CREATE TABLE excep_emps  AS SELECT * FROM employees;
```

1. Create a PL/SQL block that updates the salary of every employee to a new value of 10000 in a chosen department. Include a user-defined exception handler that handles the condition where no rows are updated and displays a custom message. Also include an exception handler that will trap any other possible error condition and display the corresponding SQLCODE and SQLERRM. Test your code three times, using department_ids 20, 30, and 40.

```
DECLARE
e_emp_salary EXCEPTION;
BEGIN
UPDATE excep_emps set salary= 10000 WHERE department_id=40;
IF SQL%NOTFOUND THEN
raise e_emp_salary;
END IF;
EXCEPTION
WHEN e_emp_salary THEN
dbms_output.put_line(SQLCODE||SQLERRM||'No se pudo actualizar el salario');
WHEN others then
dbms_output.put_line(SQLCODE||SQLERRM|| 'Error desconocido');
END;
```


2. Modify your code from question 1 to handle the condition where no rows are updated using RAISE_APPLICATION_ERROR procedure in the exception section. Use an error number of –20202. Test your code again using department_id 40 and check that the

   –20202 error is displayed.

```
DECLARE
e_emp_salary EXCEPTION;
pragma exception_init(e_emp_salary,-20202);
BEGIN
UPDATE excep_emps set salary= 10000 WHERE department_id=40;
IF SQL%NOTFOUND THEN
raise_APPLICATION_ERROR(-20202,'No se actualizo');
END IF;
```

```
EXCEPTION
WHEN e_emp_salary THEN
dbms_output.put_line(SQLCODE ||SQLERRM||'No se pudo actualizar el salario');
WHEN others then
dbms_output.put_line('Error desconocido');
END;
```

3. Modify your code from question 2 to use RAISE_APPLICATION_ERROR in the executable section instead of the exception section. Test your code again using department_id 40.

```
DECLARE
e_emp_salary EXCEPTION;
pragma exception_init(e_emp_salary,-20202);
BEGIN
UPDATE excep_emps
set salary= 10000 WHERE department_id=40;
IF SQL%NOTFOUND THEN
raise_APPLICATION_ERROR(-20202,'No se actualizo');
END IF;
EXCEPTION
WHEN e_emp_salary THEN
dbms_output.put_line(SQLERRM || SQLCODE ||' No se pudo actualizar el salario');
WHEN others then
dbms_output.put_line('Error desconocido');
END;
```

4. Be careful incorporating DELETE statements in PL/SQL blocks. If you make a mistake, you may inadvertently delete data that you didn't mean to delete.

A. Enter and run the following PL/SQL block using department_id = 40, and explain the output.

```
DECLARE
  v_dept_id     excep_emps.department_id%TYPE;
v_count       NUMBER;
BEGIN
  v_dept_id := 40;
  SELECT COUNT(*) INTO v_count
    FROM excep_emps
    WHERE department_id = v_dept_id;
  DBMS_OUTPUT.PUT_LINE('There are ' || v_count || ' employees');
  DELETE FROM excep_emps
    WHERE department_id = v_dept_id;
  DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT || ' employees were deleted');
  ROLLBACK;
END;
```

No hay ningún departamento con id 40, por eso no se elimina ningún empleado.

B. Modify your code to include two user-defined exception handlers, one to test whether SELECT returns a value of 0, and the other to test if no rows were DELETEd. Declare the exceptions and RAISE them explicitly before trapping them in the EXCEPTION section. Do NOT use RAISE_APPLICATION_ERROR. Test your modified block using department_id 40.

```
DECLARE
 v_dept_id  excep_emps.department_id%TYPE;
 v_count       NUMBER;
 e_zero EXCEPTION;
 e_deleted EXCEPTION;
BEGIN
 v_dept_id := 40;
 SELECT COUNT(*) INTO v_count
   FROM excep_emps
   WHERE department_id = v_dept_id;
 IF v_count=0 then
 raise e_zero;
end if;
 DBMS_OUTPUT.PUT_LINE('There are ' || v_count || ' employees');
 DELETE FROM excep_emps
   WHERE department_id = v_dept_id;
if SQL%NOTFOUND THEN
raise e_deleted;
END IF;
 DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT || ' employees were deleted');
 ROLLBACK;
EXCEPTION
when e_zero then
DBMS_OUTPUT.PUT_LINE('No contiene filas');
when e_deleted then
DBMS_OUTPUT.PUT_LINE('No contiene filas');
```

```
END ;
```

C. Modify your block again to use RAISE_APPLICATION_ERROR in the executable section. Use error numbers –20203 and –20204. Test your modified block using department_id 40.

```
 DECLARE
  v_dept_id  excep_emps.department_id%TYPE;
  v_count       NUMBER;
  e_zero EXCEPTION;
  pragma exception_init(e_zero,-20203);
  e_deleted EXCEPTION;
  pragma exception_init(e_deleted,-20204);

BEGIN
  v_dept_id := 40;
  SELECT COUNT(*) INTO v_count
    FROM excep_emps
    WHERE department_id = v_dept_id;
  IF v_count=0 then
raise_APPLICATION_ERROR(-20203,'No contiene filas');
end if;
  DBMS_OUTPUT.PUT_LINE('There are ' || v_count || ' employees');
  DELETE FROM excep_emps
    WHERE department_id = v_dept_id;
if SQL%NOTFOUND THEN
raise_APPLICATION_ERROR(-20204,'No se elimino nada');
END IF;
  DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT || ' employees were deleted');
  ROLLBACK;

EXCEPTION
when e_zero then
DBMS_OUTPUT.PUT_LINE('No contiene filas');

when e_deleted then
DBMS_OUTPUT.PUT_LINE('No contiene filas');

END ;
```