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.

1.
```
DECLARE
 e_no_rows_updated EXCEPTION;
BEGIN
 UPDATE excep_emps
 SET salary = 10000
 WHERE department_id = 20;
 IF SQL%ROWCOUNT = 0 THEN
 RAISE e_no_rows_updated;
 END IF;
 EXCEPTION
 WHEN e_no_rows_updated THEN
 DBMS_OUTPUT.PUT_LINE('No employees in the department');
 WHEN OTHERS THEN
 DBMS_OUTPUT.PUT_LINE(SQLCODE || '-' || SQLERRM);
END;
```

2.
```
DECLARE
 e_no_rows_updated EXCEPTION;
BEGIN
 UPDATE excep_emps
 SET salary = 10000
 WHERE department_id = 40;
 IF SQL%ROWCOUNT = 0 THEN
 RAISE e_no_rows_updated;
 END IF;
 EXCEPTION
 WHEN e_no_rows_updated THEN
 RAISE_APPLICATION_ERROR(-20202,'No employees in the department');
 WHEN OTHERS THEN
 DBMS_OUTPUT.PUT_LINE(SQLCODE || '-' || SQLERRM);
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.

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.

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                                                    There are 0 employees
  v_dept_id := 40;
  SELECT COUNT(*) INTO v_count                           0 employees were deleted
    FROM excep_emps
    WHERE department_id = v_dept_id;                     1 row(s) deleted.
  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;
```

   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.

   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.

.