

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 statement** | Use this statement to raise a named exception. |
| **User-defined error** | These errors are not automatically raısed 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\_invalid\_department EXCEPTION;

v\_deptno NUMBER :=1434556;

v\_error\_code NUMBER;

v\_error\_message VARCHAR2(255);

BEGIN

UPDATE excep\_emps

SET salary = 10000

WHERE department\_id = v\_deptno;

IF SQL%NOTFOUND THEN

RAISE e\_invalid\_department;

ROLLBACK;

v\_error\_code := SQLCODE;

v\_error\_message := SQLERRM;

END IF;

EXCEPTION

WHEN e\_invalid\_department

THEN DBMS\_OUTPUT.PUT\_LINE('Incorect department id.'||' '||v\_error\_code||' '||v\_error\_message);

END;

Copyright © 2020, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

1. 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\_invalid\_department EXCEPTION;

PRAGMA EXCEPTION\_INIT(e\_invalid\_department, -20202);

v\_deptno NUMBER := 40;

v\_error\_code NUMBER;

v\_error\_message VARCHAR2(255);

BEGIN

UPDATE excep\_emps

SET salary = 10000

WHERE department\_id = v\_deptno;

IF SQL%NOTFOUND THEN

RAISE\_APPLICATION\_ERROR(-20202, 'This is not a valid department');

END IF;

END;

1. 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

v\_deptno NUMBER := 40;

v\_error\_code NUMBER;

v\_error\_message VARCHAR2(255);

BEGIN

UPDATE excep\_emps

SET salary = 10000

WHERE department\_id = v\_deptno;

IF SQL%NOTFOUND THEN

RAISE\_APPLICATION\_ERROR(-20202, 'This is not a valid department');

END IF;

END;

1. 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;

Nu avem angajati ai departamentului 40, ca urmare avem un numar de 0 angajati stersi.

1. 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.

ECLARE

e\_no\_data EXCEPTION;

e\_no\_delete EXCEPTION;

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;

IF SQL%NOTFOUND THEN

RAISE e\_no\_data;

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\_no\_delete;

END IF;

EXCEPTION

WHEN e\_no\_data THEN DBMS\_OUTPUT.PUT\_LINE('Exception no data found.');

WHEN e\_no\_delete THEN DBMS\_OUTPUT.PUT\_LINE('Exception no delete.');

END;

1. 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.

ECLARE

e\_no\_data EXCEPTION;

e\_no\_delete EXCEPTION;

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;

EXCEPTION WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20203, 'Nu avem date');

WHEN OTHERS THEN

RAISE\_APPLICATION\_ERROR(-20204, 'Nu putem sterge');

END;