Database Programming with PL/SQL
7-1: Handling Exceptions
Practice Activities

Vocabulary

Identify the vocabulary word for each definition below:

Exception Handler	Code that defines the recovery actions to be performed when execution-time errors occur.
Exception	Occurs when an error is discovered during the execution of a program that disrupts the normal operation of the program.

Try It / Solve It

1. What happens when Oracle encounters a runtime problem while executing a PL/SQL block?

   An exception occurs when an error is discovered.

2. What do you need to add to your PL/SQL block to address these problems?

   We need to add an exception handler.

3. List three advantages of handling exceptions within a PL/SQL block.

   No more unexpected application crashes
   Protects the data from unexpected change
   The users will see our messages instead of vague error messages

4. Run this PL/SQL code and then answer the questions that follow.

```
DECLARE
  v_jobid    employees.job_id%TYPE;
BEGIN
  SELECT job_id INTO v_jobid
    FROM employees
    WHERE department_id = 80;
END;
```

B.

```
DECLARE
 v_jobid employees.job_id%TYPE;
BEGIN
 SELECT job_id INTO v_jobid
 FROM employees
 WHERE department_id = 80;
 EXCEPTION
 WHEN TOO_MANY_ROWS THEN
 DBMS_OUTPUT.PUT_LINE('The statement returns more than a row');
END;
```

   A. What happens when you run the block? In your own words, explain what you can do to fix this problem.

   ORA-01422: exact fetch returns more than requested number of rows

   The statement returns more than one row and an exception appears so we need to handle it.

   B. Modify the code to fix the problem. Use a TOO_MANY_ROWS exception handler.

   C. Run your modified code. What happens this time?

   The statement returns more than a row
   Statement processed

5. Run the following PL/SQL block, which tries to insert a new row (with department_id = 50) into the departments table. What happens and why?

```
BEGIN
  INSERT INTO departments (department_id, department_name,
              manager_id, location_id)
    VALUES (50, 'A new department', 100, 1500);
  DBMS_OUTPUT.PUT_LINE('The new department was inserted');
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE ('An exception has occurred.');
END;
```

An exception has occurred.

1 row(s) inserted.

The exception occured at insert so the execution jumped to the exceptions block. The rest of code was ignored.

6. **Enter the following code to create a copy of the employees table for this and the next question.**

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

In the new emp_temp table, delete all but one of the employees in department 10.

```
SELECT * FROM emp_temp WHERE department_id = 10;

DELETE FROM emp_temp WHERE employee_id = …; (repeat as necessary)
```

Enter the following PL/SQL block, which tries to SELECT all the employees in a specific department. Run it three times, using department_ids 10, 20, and 30. What happens and why?

```
DECLARE
  v_employee_id   emp_temp.employee_id%TYPE;
  v_last_name     emp_temp.last_name%TYPE;
BEGIN
  SELECT employee_id, last_name INTO v_employee_id, v_last_name
    FROM emp_temp
    WHERE department_id = 10;   -- run with values 10, 20, and 30
  DBMS_OUTPUT.PUT_LINE('The SELECT was successful');
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An exception has occurred');
END;
```

The select was succesfull for department 10

For department 20 and 30, An exception has occurred (but we don't know many details about it) WHY?

7. Modify your code from question 6 to add two more exception handlers to trap the possible exceptions individually. Use NO_DATA_FOUND and TOO_MANY_ROWS. Re-run the block three times, using 10, 20, and 30 as before. Observe the message displayed in each case.

When finished, remember to delete the emp_temp table.

```
DROP TABLE emp_temp;
```

8. List three guidelines for trapping exceptions.

Write out debugging information in your exception handlers

Carefully consider whether each exception handler should commit the transaction, roll it back, or let it continue

No matter how severe the error is, you want to leave the database in a consistent state and avoid storing any bad data

9. Enter and run the following PL/SQL block. Explain the output. Note: the WHEN OTHERS handler successfully handles any type of exception which occurs.

```
DECLARE
  v_number     NUMBER(2);
BEGIN
  v_number := 9999;
EXCEPTION
 WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('An exception has occurred');
END;
```

An exception has occurred

Statement processed.

9999 has more than 2 characters (4 characters in number(2) results in an error)
The error was handled

10. Modify the block in question 9 to omit the exception handler, then re-run the block. Explain the output.

```
DECLARE
 v_number NUMBER(4);
BEGIN
 v_number := 9999;
EXCEPTION
 WHEN OTHERS THEN
 DBMS_OUTPUT.PUT_LINE('An exception has occurred');
END;
```

Statement processed.
The variable gets the value but nothing will print because the exception handle code will be ignored.

11. Enter and run the following code and explain the output.

```
DECLARE
  v_number    NUMBER(4);
BEGIN
  v_number := 1234;
    DECLARE
      v_number     NUMBER(4);
    BEGIN
      v_number := 5678;
      v_number := 'A character string';
    END;
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An exception has occurred');
    DBMS_OUTPUT.PUT_LINE('The number is: ' || v_number);
END;
```

An exception has occurred

The number is: 1234

Statement processed.

The OTHERS handler traps all the exceptions that are not trapped and the number 1234 is the first to be executed and not considered trapped.