Exercises for Chapter 9: Exceptions

The Labs below provide you with exercises and suggested answers with discussion related to how those answers resulted. The most important thing to realize is whether your answer works. You should figure out the implications of the answers here and what the effects are from any different answers you may come up with.

Lab 9.1 Exception Scope

In this exercise, you will determine whether a value of ZIP code provided at the run time is valid and then display name of an instructor that resides in this particular ZIP code. You will use nested PL/SQL blocks to achieve the desired results. The original PL/SQL script does not contain any exception handlers. Therefore, you will be asked to identify run-time errors that may occur and define exception handlers for them.

Create the following PL/SQL script:

For Example *ch09_8a.sql*

```
<<outer_block>>
DECLARE
   v_zip   VARCHAR2(5) := '&sv_zip';
   v_name  VARCHAR2(50);

BEGIN
   DBMS_OUTPUT.PUT_LINE ('Check if provided zipcode is valid');
   SELECT zip
     INTO v_zip
     FROM zipcode
    WHERE zip = v_zip;

   <<inner_block>>
   BEGIN
      SELECT first_name||' '||last_name
        INTO v_name
        FROM instructor
       WHERE zip = v_zip;

      DBMS_OUTPUT.PUT_LINE ('Instructor name is '||v_name);
   END;
END;
```

In order to test this script fully, execute it three times for the following ZIP code values: 10005, 10015, and 00914, and then answer the following questions:

a) What output was generated by the script (for all values of ZIP)?

   Answer: The script generated these three outputs for the specified values of ZIP.

   For value of ZIP code 10005 the output is as follows:

```
Check if provided zipcode is valid
Instructor name is Marilyn Frantzen
```

   For value of ZIP code 10015 the output is as shown:

```
Check if provided zipcode is valid
ORA-01422: exact fetch returns more than requested number of rows
ORA-06512: at line 16
```

   For the value of ZIP code 00914 the output is as follows:

```
Check if provided zipcode is valid
ORA-01403: no data found
ORA-06512: at line 16
```

b) Explain the output produced by the example for all three values of ZIP code.

   Answer: For the first value of ZIP code 10005, the script is able to execute successfully. First, the value of ZIP code provided at the run time is checked against the `ZIPCODE` table and the message 'Check if provided zipcode is valid' is displayed in the Dbms Output window. Next, instructor's name is selected from the `INSTRUCTOR` table and displayed in the Dbms Output window as well.

   For the second value of ZIP code 10015, the script is not able to complete successfully. Similarly, to the first run, the value of the ZIP code provided at the run time is evaluated and the results are displayed in the Dbms Output window. However, when the instructor name is selected from the `INSTRUCTOR` table, the error is encountered because there is more than one instructor for this ZIP code. As a result, the error message 'exact fetch returns…' is displayed in the script output window.

   Finally, for the third value of ZIP code 00914, the script also fails to complete successfully because there are no records in the `INSTRUCTOR` table corresponding to this value of ZIP code. In this case, the error message 'no data found' is displayed in the script output window.

c) Based on the errors encountered above, what exception handlers must be added to the script?

   Answer: The modified script should look similar to one of the following versions. Based on the error messages encountered by the original script, `TOO_MANY_ROWS` and `NO_DATA_FOUND` exceptions must be added to the script. Newly added exception section is shown in bold.

   For Example *ch09_8b.sql*

```
<<outer_block>>
DECLARE
   v_zip   VARCHAR2(5) := '&sv_zip';
   v_name  VARCHAR2(50);

BEGIN
   DBMS_OUTPUT.PUT_LINE ('Check if provided zipcode is valid');
   SELECT zip
```

```
     INTO v_zip
     FROM zipcode
    WHERE zip = v_zip;

   <<inner_block>>
   BEGIN
      SELECT first_name||' '||last_name
        INTO v_name
        FROM instructor
       WHERE zip = v_zip;

      DBMS_OUTPUT.PUT_LINE ('Instructor name is '||v_name);
   EXCEPTION
      WHEN TOO_MANY_ROWS
      THEN
         DBMS_OUTPUT.PUT_LINE
            ('More than one instructor resides in this zip code');
      WHEN NO_DATA_FOUND
      THEN
         DBMS_OUTPUT.PUT_LINE
            ('There are no instructors residing in this zip code');
   END;
END;
```

For Example *ch09_8c.sql*

```
<<outer_block>>
DECLARE
   v_zip   VARCHAR2(5) := '&sv_zip';
   v_name  VARCHAR2(50);

BEGIN
   DBMS_OUTPUT.PUT_LINE ('Check if provided zipcode is valid');
   SELECT zip
     INTO v_zip
     FROM zipcode
    WHERE zip = v_zip;

   <<inner_block>>
   BEGIN
      SELECT first_name||' '||last_name
        INTO v_name
        FROM instructor
       WHERE zip = v_zip;

      DBMS_OUTPUT.PUT_LINE ('Instructor name is '||v_name);
   END;
EXCEPTION
   WHEN TOO_MANY_ROWS
   THEN
      DBMS_OUTPUT.PUT_LINE
```

```
            ('More than one instructor resides in this zip code');
      WHEN NO_DATA_FOUND
      THEN
         DBMS_OUTPUT.PUT_LINE
            ('There are no instructors residing in this zip code');
   END;
```

Note that both versions of the script have been expanded with the exception-handling section. In the second version of the script (ch09_8b.sql), the exception handling section is added to the inner block. Whereas, in the third version of script (ch09_8c.sql), the exception handling section is added to the outer block. Both versions of the script are similar in their behavior of catching the error and terminating successfully as illustrated by the output below:

For value of ZIP code 10015 the output is as shown:

```
Check if provided zipcode is valid
More than one instructor resides in this zip code
```

For the value of ZIP code 00914 the output is as follows:

```
Check if provided zipcode is valid
There are no instructors residing in this zip code
```

Lab 9.2 User-Defined Exceptions

In this exercise, you will define an exception that will allow you to raise an error if an instructor teaches ten or more sections. Create the following PL/SQL script:

For Example *ch09_9a.sql*

```
DECLARE
   v_instructor_id     NUMBER := &sv_instructor_id;
   v_tot_sections      NUMBER;
   v_name              VARCHAR2(30);
   e_too_many_sections EXCEPTION;
BEGIN
   SELECT RTRIM(first_name)||' '||RTRIM(last_name)
     INTO v_name
     FROM instructor
    WHERE instructor_id = v_instructor_id;

   SELECT COUNT(*)
     INTO v_tot_sections
     FROM section
    WHERE instructor_id = v_instructor_id;

   IF v_tot_sections >= 10
   THEN
      RAISE e_too_many_sections;
   ELSE
      DBMS_OUTPUT.PUT_LINE
         ('Instructor, '||v_name||', teaches '||v_tot_sections||' sections');
   END IF;
EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
```

```
      DBMS_OUTPUT.PUT_LINE ('This is not a valid instructor');

   WHEN e_too_many_sections
   THEN
      DBMS_OUTPUT.PUT_LINE ('Instructor, '||v_name||', teaches too much');
END;
```

Execute the script twice providing 101 and 102 for the values of instructor ID, and then answer the following questions:

a) What output was generated by the script these instructor IDs? Explain the difference in the outputs produced.

   Answer: The script generated these two outputs for the specified values of instructor ID.

   For the value of instructor ID 101 the output is as follows:

   ```
   Instructor, Fernand Hanks, teaches 9 sections
   ```

   For the value of instructor ID 102 the output is as follows:

   ```
   Instructor, Tom Wojick, teaches too much
   ```

   The first output is produced when value 101 is provided for the instructor ID. Since the number of sections taught by this instructor is less than 10, the `ELSE` part of the `IF-THEN-ELSE` statement is executed, and the instructor's name is displayed on the screen.

   The second output is produced when value 102 is provided for the instructor ID. In this case, number of sections taught by the instructor is 10. As a result, the `IF` part of the `IF-THEN-ELSE` statement is executed, and user-defined exception is raised. Once the exception is raised, the control of the execution is transferred to the exception-handling section of the block, and the exception message is displayed in the Dbms Output window.

b) What is the condition that causes the user-defined exception to be raised?

   Answer: The user-defined exception is raised if the condition

   ```
   v_tot_sections >= 10
   ```

   evaluates to `TRUE`. In other words, if an instructor teaches ten or more sections, the exception `e_too_many_sections` is raised.

Lab 9.3 Exception Propagation

Answer the following questions.

Exception Propagation

In this exercise, you will use nested PL/SQL blocks to practice exception propagation. You will be asked to experiment with the script via exceptions. Try to answer the questions before you run the script. Once you have answered the questions, run the script and check your answers.

Create the following PL/SQL script:

For Example *ch09_10a.sql*

```
DECLARE
   v_my_name VARCHAR2(15) := 'THIS IS A REALLY LONG NAME';
BEGIN
```

```
   DBMS_OUTPUT.PUT_LINE ('My name is '||v_my_name);

   DECLARE
      v_your_name VARCHAR2(15);
   BEGIN
      v_your_name := '&sv_your_name';
      DBMS_OUTPUT.PUT_LINE ('Your name is '||v_your_name);
   EXCEPTION
      WHEN VALUE_ERROR
      THEN
         DBMS_OUTPUT.PUT_LINE ('Error in the inner block');
         DBMS_OUTPUT.PUT_LINE ('This name is too long');
   END;

EXCEPTION
   WHEN VALUE_ERROR
   THEN
      DBMS_OUTPUT.PUT_LINE ('Error in the outer block');
      DBMS_OUTPUT.PUT_LINE ('This name is too long');
END;
```

Answer the following questions first, and then execute the script:

a) What exception is raised by the assignment statement in the declaration section of the outer block?

   Answer: The exception VALUE_ERROR is raised by the assignment statement of the outer block. The variable v_my_name is declared as VARCHAR2(15). However, the value that is assigned to this variable contains 26 characters. As a result, the assignment statement causes a run time error.

b) Once this exception (based on the previous question) is raised, will the program be able to terminate successfully? Explain why or why not.

   Answer: When that exception VALUE_ERROR is raised, the script is not able to complete successfully because the error occurs in the declaration section of the outer block. Since the outer block is not enclosed by any other block, control is transferred to the host environment. As a result, an error message is generated when this example is run.

c) How would you change this script so that the exception section in the outer block is able to handle an error caused by the assignment statement in the outer block?

   Answer: In order for the exception section of the outer block to be able to handle the error generated by the assignment statement, the assignment statement must be moved to the executable section of this block. All changes are shown in bold.

For Example *ch09_10b.sql*

```
DECLARE
   v_my_name VARCHAR2(15);
BEGIN
   v_my_name := 'THIS IS A REALLY LONG NAME';
   DBMS_OUTPUT.PUT_LINE ('My name is '||v_my_name);
   DECLARE
      v_your_name VARCHAR2(15);
   BEGIN
      v_your_name := '&sv_your_name';
      DBMS_OUTPUT.PUT_LINE ('Your name is '||v_your_name);
   EXCEPTION
      WHEN VALUE_ERROR
      THEN
         DBMS_OUTPUT.PUT_LINE ('Error in the inner block');
         DBMS_OUTPUT.PUT_LINE ('This name is too long');
   END;

EXCEPTION
   WHEN VALUE_ERROR
   THEN
      DBMS_OUTPUT.PUT_LINE ('Error in the outer block');
      DBMS_OUTPUT.PUT_LINE ('This name is too long');
END;
```

The new version of this script produces the following output:

```
Error in the outer block
This name is too long
```

d) Change the value of the variable from 'THIS IS A REALLY LONG NAME' to 'MY NAME'. Then change the script so that if there is an error caused by the assignment statement of the inner block, it is handled by the exception-handling section of the outer block.

Answer: The script should look similar to the script below. All changes are highlighted in bold.

For Example *ch09_10c.sql*

```
DECLARE
   v_my_name VARCHAR2(15) := 'MY NAME';
BEGIN
   DBMS_OUTPUT.PUT_LINE ('My name is '||v_my_name);

   DECLARE
      v_your_name VARCHAR2(15) := '&sv_your_name';
   BEGIN
      DBMS_OUTPUT.PUT_LINE ('Your name is '||v_your_name);
   EXCEPTION
      WHEN VALUE_ERROR
      THEN
         DBMS_OUTPUT.PUT_LINE ('Error in the inner block');
         DBMS_OUTPUT.PUT_LINE ('This name is too long');
   END;

EXCEPTION
   WHEN VALUE_ERROR
   THEN
      DBMS_OUTPUT.PUT_LINE ('Error in the outer block');
      DBMS_OUTPUT.PUT_LINE ('This name is too long');
```

```
END;
```

In this version of the example, the assignment statement was moved from the executable section of the inner block to the declaration section of the same block. As a result, if an exception is raised by this assignment statement, control of the execution is transferred to the exception section of the outer block.

Next, consider another option where the exception caused by the assignment statement of the inner block is caught by the exception section of the inner block and then re-raised in the outer block.

For Example *ch09_10d.sql*

```
DECLARE
   v_my_name VARCHAR2(15) := 'MY NAME';
BEGIN
   DBMS_OUTPUT.PUT_LINE ('My name is '||v_my_name);

   DECLARE
      v_your_name VARCHAR2(15);
   BEGIN
      v_your_name := '&sv_your_name';
      DBMS_OUTPUT.PUT_LINE ('Your name is '||v_your_name);
   EXCEPTION
      WHEN VALUE_ERROR
      THEN
         RAISE;
   END;

EXCEPTION
   WHEN VALUE_ERROR
   THEN
      DBMS_OUTPUT.PUT_LINE ('Error in the outer block');
      DBMS_OUTPUT.PUT_LINE ('This name is too long');
END;
```

In this version of the example, the `RAISE` statement was used in the exception-handling section of the inner block. As a result, the exception is re-raised in the outer block.

Both versions of this example produce output as shown below:

```
My name is MY NAME
Error in the outer block
This name is too long
```

Re-raising Exceptions

In this exercise, you will check the number of sections for a given course. If a course does not have a section associated with it, you will raise a user-defined exception, `e_no_sections`. Again, try to answer the questions before you run the script. Once you have answered the questions, run the script and check your answers.

Create the following PL/SQL script:

For Example *ch09_11a.sql*

```
DECLARE
   v_course_no   NUMBER := 430;
   v_total       NUMBER;
   e_no_sections EXCEPTION;
BEGIN
   BEGIN
      SELECT COUNT(*)
        INTO v_total
        FROM section
       WHERE course_no = v_course_no;

      IF v_total = 0
      THEN
         RAISE e_no_sections;
      ELSE
         DBMS_OUTPUT.PUT_LINE ('Course, '||v_course_no||' has '||v_total||' sections');
      END IF;
   EXCEPTION
      WHEN e_no_sections
      THEN
         DBMS_OUTPUT.PUT_LINE ('There are no sections for course '||v_course_no);
   END;
END;
```

Answer the following questions first, and then execute the script:

a) What exception will be raised if there are no sections for a given course number?

   Answer: If there are no sections for a given course number, the exception `e_no_sections` is raised.

b) If the exception `e_no_sections` is raised, how will the control of the execution flow? Explain your answer.

   Answer: If the exception `e_no_sections` is raised, the control of the execution will be passed from the inner block to the exception-handling section of that inner block. This is possible because the inner block has exception-handling section, in which the exception is raised and handled.

c) Change this script so that the exception `e_no_sections` is re-raised in the outer block.

   Answer: Your script should look similar to the following. All changes are highlighted in bold.

   For Example *ch09_11b.sql*

```
DECLARE
   v_course_no   NUMBER := 430;
   v_total       NUMBER;
   e_no_sections EXCEPTION;
BEGIN
   BEGIN
      SELECT COUNT(*)
        INTO v_total
        FROM section
       WHERE course_no = v_course_no;
      IF v_total = 0
      THEN
         RAISE e_no_sections;
      ELSE
         DBMS_OUTPUT.PUT_LINE ('Course, '||v_course_no||' has '||v_total||'
sections');
      END IF;
   EXCEPTION
      WHEN e_no_sections
      THEN
         RAISE;
   END;

EXCEPTION
   WHEN e_no_sections
   THEN
      DBMS_OUTPUT.PUT_LINE ('There are no sections for course '||v_course_no);
END;
```

In this version of the example, the exception-handling section of the inner block was modified. The `DBMS_OUTPUT.PUT_LINE` statement has been replaced by the `RAISE` statement. In addition, the exception-handling section was included in the outer block.

Try It Yourself

The projects in this section are meant to have you use all of the skills that you have acquired throughout this chapter. Here are some exercises that will help you test the depth of your understanding.

1) Create the following script. For a course section provided at a run time determine the number of students registered. If this number is equal to or greater than 10, raise the user-defined exception `e_too_many_students` and display the error message. Otherwise, display how many students are in this section.

   Answer: The script should look similar to the following:

   For Example *ch09_12a.sql*

```
DECLARE
   v_section_id        NUMBER := &sv_section_id;
   v_total_students    NUMBER;
   e_too_many_students EXCEPTION;
BEGIN
   -- Calculate number of students enrolled
   SELECT COUNT(*)
     INTO v_total_students
     FROM enrollment
    WHERE section_id = v_section_id;

   IF v_total_students >= 10
   THEN
      RAISE e_too_many_students;
   ELSE
```

```
      DBMS_OUTPUT.PUT_LINE
         ('There are '||v_total_students||' students in section '||v_section_id);
   END IF;
EXCEPTION
   WHEN e_too_many_students
   THEN
      DBMS_OUTPUT.PUT_LINE
         ('There are too many students in section '||v_section_id);
END;
```

The script above declares two variables, `v_section_id` and `v_total_students`, to store section ID provided by a user and total number of students enrolled in that section respectively. It also declares a user-defined exception `e_too_many_students`.

The executable portion of the script checks how many students are enrolled in a given section and raises the exception `e_too_many_students` via the `IF-THEN` statement if the value returned by the `COUNT` function equals to or exceeds 10, or displays the message specifying how many students are enrolled in a given section.

To test this script fully, consider running it for three values of section ID. When 101 is provided for the section ID (this section has more than 10 students), this script produces output as follows:

```
There are too many students in section 101
```

When 116 is provided for the section ID (this section has less than 10 students), this script produces different output:

```
There are 8 students in section 116
```

Next, consider running this script for non-existent section ID as follows:

```
There are 0 students in section 999
```

Note the script did not produce any errors. Instead it stated that section 999 has 0 students in it. How would you modify this script to ensure that when there is no corresponding section ID in the `ENROLLMENT` table, a message “This section does not exist” is displayed on the screen?

2) Restructure the script you created in the previous exercise so that it has nested PL/SQL blocks. Once the exception `e_too_many_students` has been raised in the inner block, re-raise it in the outer block.

Answer: The new version of the script should look similar to the following. Affected statements are shown in bold.

For Example *ch09_12b.sql*

```
DECLARE
   v_section_id        NUMBER := &sv_section_id;
   v_total_students    NUMBER;
   e_too_many_students EXCEPTION;
BEGIN
   -- Add inner block
   BEGIN
      -- Calculate number of students enrolled
      SELECT COUNT(*)
```

```
         INTO v_total_students
         FROM enrollment
        WHERE section_id = v_section_id;

      IF v_total_students >= 10
      THEN
         RAISE e_too_many_students;
      ELSE
         DBMS_OUTPUT.PUT_LINE
            ('There are '||v_total_students||' students in section '||
             v_section_id);
      END IF;
   -- Re-raise exception
   EXCEPTION
      WHEN e_too_many_students
      THEN
         RAISE;
   END;
EXCEPTION
   WHEN e_too_many_students
   THEN
      DBMS_OUTPUT.PUT_LINE
         ('There are too many students in section '||v_section_id);
END;
```

This version of the script, contains inner block where `e_too_many_students` exception is raised first and then propagated to the outer block. This version of the script produces output identical to the original script.

Next, consider a different version where the original PL/SQL block (PL/SQL block from the original script) has been enclosed by another block as shown:

For Example *ch09_12c.sql*

```
-- Outer PL/SQL block
BEGIN
   -- This block became inner PL/SQL block
   DECLARE
      v_section_id        NUMBER := &sv_section_id;
      v_total_students    NUMBER;
      e_too_many_students EXCEPTION;
   BEGIN
      -- Calculate number of students enrolled
      SELECT COUNT(*)
        INTO v_total_students
        FROM enrollment
       WHERE section_id = v_section_id;

      IF v_total_students >= 10
      THEN
         RAISE e_too_many_students;
      ELSE
         DBMS_OUTPUT.PUT_LINE
            ('There are '||v_total_students||' students in section '||
            v_section_id);
      END IF;
   EXCEPTION
      WHEN e_too_many_students
      THEN
         RAISE;
   END;

EXCEPTION
   WHEN e_too_many_students
   THEN
      DBMS_OUTPUT.PUT_LINE
         ('There are too many students in section '||v_section_id);
END;
```

This version of the script causes the following error message:

```
ORA-06550: line 28, column 9:
PLS-00201: identifier 'E_TOO_MANY_STUDENTS' must be declared
```

This occurs because the exception `e_too_many_students` is declared in the inner block and as a result is not visible to the outer block. In addition, the `v_section_id` variable used by the exception-handling section of the outer block is declared in the inner block as well, and as a result, is not accessible in the outer block.

To correct these errors, this version script can be modified as follows:

For Example *ch09_12d.sql*

```
-- Outer PL/SQL block
DECLARE
   v_section_id        NUMBER := &sv_section_id;
   e_too_many_students EXCEPTION;
BEGIN
   -- This block became inner PL/SQL block
   DECLARE
      v_total_students NUMBER;
   BEGIN
      -- Calculate number of students enrolled
      SELECT COUNT(*)
        INTO v_total_students
        FROM enrollment
       WHERE section_id = v_section_id;

      IF v_total_students >= 10
      THEN
         RAISE e_too_many_students;
      ELSE
         DBMS_OUTPUT.PUT_LINE
            ('There are '||v_total_students||' students in section '||
             v_section_id);
      END IF;
   EXCEPTION
      WHEN e_too_many_students
      THEN
         RAISE;
   END;

EXCEPTION
   WHEN e_too_many_students
   THEN
      DBMS_OUTPUT.PUT_LINE
         ('There are too many students for section '||v_section_id);
END;
```