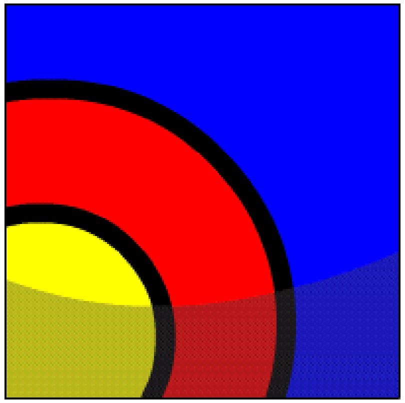
Trapping Oracle Server Exceptions

What Will I Learn?

In this lesson, you will learn to:

- Describe and provide an example of an error defined by the Oracle server.
- Describe and provide an example of an error defined by the PL/SQL programmer
- Differentiate between errors that are handled implicitly and explicitly by the Oracle server
- Write PL/SQL code to trap a predefined Oracle server error
- Write PL/SQL code to trap a non-predefined Oracle server error
- Write PL/SQL code to identify an exception by error code and by error message

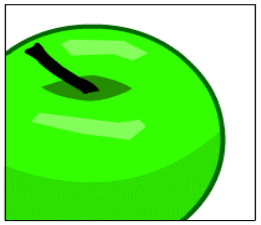


Why Learn It?

PL/SQL error handling is flexible and allows programmers to use both errors defined by the Oracle server and errors defined by the programmer.

This lesson discusses predefined and non-predefined Oracle server errors. Predefined errors are the common Oracle errors for which PL/SQL has predefined exception names. Non-predefined errors make use of the ORA error codes and messages. The syntax is different for each, but you can trap both kinds of errors in the EXCEPTION section of your PL/SQL program.

Tell Me / Show Me



Exception Types

Exception	Description	Instructions for Handling
Predefined Oracle server error	One of approximately 20 errors that occur most often in PL/SQL code	You need not declare these exceptions. They are predefined by the Oracle server and are raised implicitly (automatically).
Non-predefined Oracle server error	Any other standard Oracle server error	Declare within the declarative section and allow the Oracle Server to raise them implicitly (automatically).
User-defined error	A condition that the PL/SQL programmer decides is abnormal	Declare within the declarative section, and raise explicitly.

Tell Me / Show Me

Handling Exceptions with PL/SQL

There are two methods for raising an exception

- Implicitly (automatically) by the Oracle server: An Oracle error occurs and the associated exception is raised automatically. For example, if the error `ORA-01403` occurs when no rows are retrieved from the database in a `SELECT` statement, then PL/SQL raises the exception `NO_DATA_FOUND`.
- Explicitly by the programmer: Depending on the business functionality your program is implementing, you might have to explicitly raise an exception. You raise an exception explicitly by issuing the `RAISE` statement within the block. The exception being raised can be either user-defined or predefined. These are explained in the next lesson.

Tell Me / Show Me

Two Types of Oracle Server Error

When an Oracle server error occurs, the Oracle server automatically raises the associated exception, skips the rest of the executable section of the block, and looks for a handler in the exception section. There are two types of Oracle server errors:

- Predefined Oracle server errors: Each of these errors has a predefined name. For example, if the error `ORA-01403` occurs when no rows are retrieved from the database in a `SELECT` statement, then PL/SQL raises the predefined exception-name `NO_DATA_FOUND`.
- Non-predefined Oracle server errors: Each of these errors has a standard Oracle error number (ORA-nnnnn) and error message, but not a predefined name. You declare your own names for these so that you can reference these names in the exception section.

Tell Me / Show Me

Trapping Predefined Oracle Server Errors

- Reference the predefined name in the exception handling routine.
- Sample predefined exceptions:
  - `NO_DATA_FOUND`
  - `TOO_MANY_ROWS`
  - `INVALID_CURSOR`
  - `ZERO_DIVIDE`
  - `DUP_VAL_ON_INDEX`
- For a partial list of predefined exceptions, refer to the short list available from the Student Resources in Section 0. For a complete list of predefined exceptions, see the *PL/SQL User's Guide and Reference*.

Tell Me / Show Me

Trapping Predefined Oracle Server Errors

The following example uses the TOO_MANY_ROWS predefined Oracle server error. Note that it is not declared in the DECLARATION section.

```
DECLARE
  v_lname VARCHAR2(15);
BEGIN
  SELECT last_name INTO v_lname
    FROM employees WHERE job_id = 'ST_CLERK';
  DBMS_OUTPUT.PUT_LINE('The last name of the ST_CLERK is : '||v_lname);
EXCEPTION
  WHEN TOO_MANY_ROWS THEN
    DBMS_OUTPUT.PUT_LINE (' Your select statement retrieved multiple rows. Consider using a cursor.');
END;
```

Tell Me / Show Me

Trapping Several Predefined Oracle Server Errors

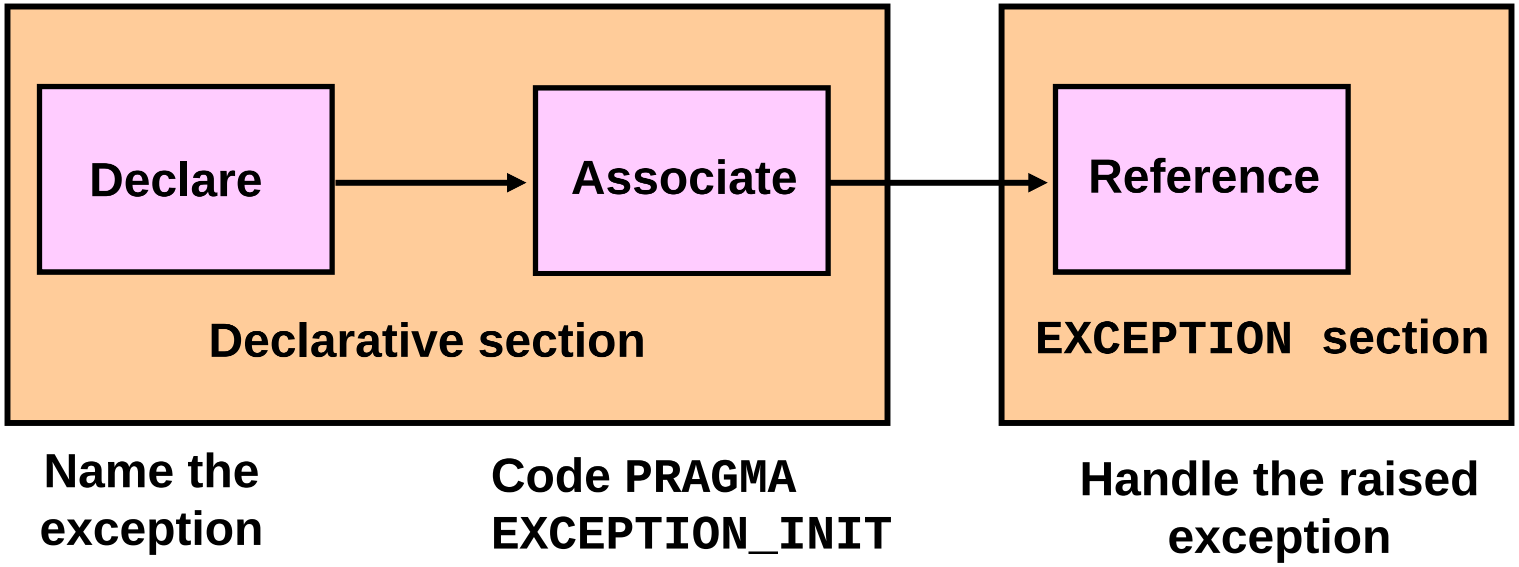
This example handles TOO_MANY_ROWS and NO_DATA_FOUND, with an OTHERS handler in case any other error occurs.

```
DECLARE
  v_lname VARCHAR2(15);
BEGIN
  SELECT last_name INTO v_lname
    FROM employees WHERE job_id = 'ST_CLERK';
  DBMS_OUTPUT.PUT_LINE('The last name of the ST_CLERK is : '||v_lname);
EXCEPTION
 WHEN TOO_MANY_ROWS THEN
    DBMS_OUTPUT.PUT_LINE ('Select statement found multiple rows');
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE ('Select statement found no rows');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE ('Another type of error occurred');
END;
```

Tell Me / Show Me

Trapping Non-Predefined Oracle Server Errors

Non-predefined exceptions are similar to predefined exceptions; however, they do not have predefined names in PL/SQL. They are standard Oracle server errors and have ORA- error numbers. You create your own names for them in the `DECLARE` section and associate these names with ORA- error numbers using the `PRAGMA EXCEPTION_INIT` function.



Tell Me / Show Me

Trapping Non-Predefined Oracle Server Errors (continued)

- You can trap a non-predefined Oracle server error by declaring it first. The declared exception is raised implicitly. In PL/SQL, the `PRAGMA EXCEPTION_INIT` tells the compiler to associate an exception name with an Oracle error number.

- This allows you to refer to any Oracle Server exception by name and to write a specific handler for it.

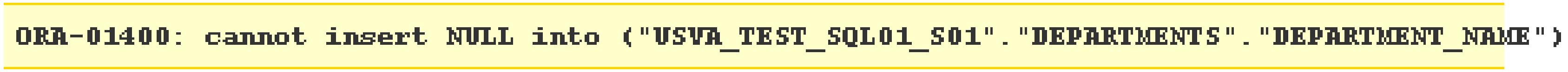
ORACLE Academy

Tell Me / Show Me

Non-Predefined Error

Examine the following example.

```
BEGIN
 INSERT INTO departments
   (department_id, department_name) VALUES (280, NULL);
END;
```



```
ORA-01400: cannot insert NULL into ("USVA_TEST_SQL01_S01"."DEPARTMENTS"."DEPARTMENT_NAME")
```

The `INSERT` statement tries to insert the value `NULL` for the `department_name` column of the `departments` table. However, the operation is not successful because `department_name` is a `NOT NULL` column. There is no predefined error name for violating a `NOT  NULL` constraint. The way to work around this problem is to declare you own name and associate it with the ORA-01400 error.

Tell Me / Show Me

Non-Predefined Error (continued)

1. Declare the name of the exception in the declarative section.

```
DECLARE
  e_insert_excep EXCEPTION;                1
  PRAGMA EXCEPTION_INIT
      (e_insert_excep, -01400);            2
BEGIN
  INSERT INTO departments
    (department_id, department_name)
    VALUES (280, NULL);
EXCEPTION
  WHEN e_insert_excep                      3
    THEN
      DBMS_OUTPUT.PUT_LINE('INSERT FAILED');
END;
```

Syntax:

exception name EXCEPTION;

where EXCEPTION is the name of the exception

Tell Me / Show Me

Non-Predefined Error (continued)

2. Associate the declared exception with the standard Oracle server error number using the PRAGMA EXCEPTION_INIT function.

```
DECLARE
  e_insert_excep EXCEPTION;            ← 1
  PRAGMA EXCEPTION_INIT
      (e_insert_excep, -01400);        ← 2
BEGIN
  INSERT INTO departments
  (department_id, department_name)
    VALUES (280, NULL);
EXCEPTION
  WHEN e_insert_excep                  ← 3
    THEN
      DBMS_OUTPUT.PUT_LINE('INSERT FAILED');
END;
```

`PRAGMA EXCEPTION_INIT (exception, error_number);`

where `exception` is the previously declared exception name and `error_number` is a standard Oracle server error number, including the hyphen in front of it.

ORACLE Academy

Tell Me / Show Me

Non-Predefined Error (continued)

3. Reference the declared exception name within the corresponding exception-handling routine.

```
DECLARE
  e_insert_excep EXCEPTION;                    ← 1
  PRAGMA EXCEPTION_INIT
      (e_insert_excep, -01400);                ← 2
BEGIN
  INSERT INTO departments
  (department_id, department_name)
    VALUES (280, NULL);
EXCEPTION
  WHEN e_insert_excep                          ← 3
    THEN
      DBMS_OUTPUT.PUT_LINE('INSERT FAILED');
END;
```

Tell Me / Show Me

Functions for Trapping Exceptions

When an exception occurs, you can retrieve the associated error code or error message by using two functions. Based on the values of the code or the message, you can decide which subsequent actions to take.

- `SQLERRM` returns character data containing the message associated with the error number.
- `SQLCODE` returns the numeric value for the error code. (You can assign it to a `NUMBER` variable.)

SQLCODE Value	Description
0	No exception encountered
1	User defined exception
+100	`NO_DATA_FOUND` exception
Negative number	Another Oracle Server error number

ORACLE Academy

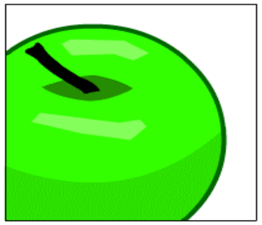
Tell Me / Show Me

Functions for Trapping Exceptions (continued)

You cannot use SQLCODE or SQLERRM directly in an SQL statement. Instead, you must assign their values to local variables, then use the variables in the SQL statement, as shown in the following example:

```
DECLARE
  v_error_code      NUMBER;
  v_error_message   VARCHAR2(255);
BEGIN
...
EXCEPTION
  WHEN OTHERS THEN
    ROLLBACK;
    v_error_code    := SQLCODE ;
    v_error_message := SQLERRM ;
    INSERT INTO error_log(e_user,e_date,error_code,error_message)
                   VALUES(USER,SYSDATE,v_error_code,v_error_message);
END;
```

Tell Me / Show Me



Terminology

Key terms used in this lesson include:

Predefined Oracle server errors

Non-predefined Oracle server errors

`PRAGMA EXCEPTION_INIT`

`SQLERRM`

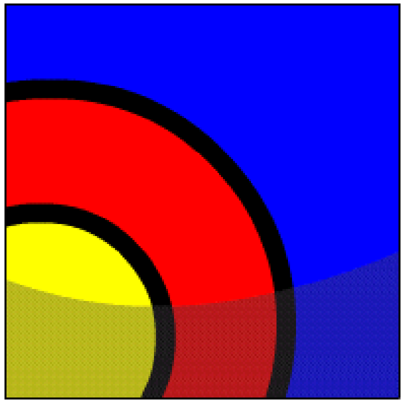
`SQLCODE`

ORACLE Academy

Summary

In this lesson, you learned to:

- Describe and provide an example of an error defined by the Oracle server.
- Describe and provide an example of an error defined by the PL/SQL programmer
- Differentiate between errors that are handled implicitly and explicitly by the Oracle server
- Write PL/SQL code to trap a predefined Oracle server error
- Write PL/SQL code to trap a non-predefined Oracle server error
- Write PL/SQL code to identify an exception by error code and by error message



Try It / Solve It

The exercises in this lesson cover the following topics:

- Listing and describing different types of PL/SQL exception handlers
- Differentiating between errors that are handled implicitly and explicitly by the Oracle server
- Trapping predefined Oracle server errors
- Trapping non-predefined Oracle server errors