

Department of Software Engineering  
Mehran University of Engineering and Technology, Jamshoro

Course: SW222 – Database Management & Administration

<b>Instructor</b>	Ms Shafiya Qadeer	<b>Practical/Lab No.</b>	13
<b>Date</b>	2021	<b>CLOs</b>	3
<b>Signature</b>		<b>Assessment Score</b>	2 Marks

**Topic** To become familiar with Exception Handling

**Objectives** - To become familiar with Pre-defined & User Defined Exceptions

**Lab Discussion: Theoretical concepts and Procedural steps**

**Exception:**

An exception is an error condition during a program execution. PL/SQL supports programmers to catch such conditions using **EXCEPTION** block in the program and an appropriate action is taken against the error condition. There are two types of exceptions –

- System-defined exceptions
- User-defined exceptions

**Syntax for Exception Handling**

The general syntax for exception handling is as follows. Here you can list down as many exceptions as you can handle. The default exception will be handled using ***WHEN others THEN*** –

```
DECLARE

<declarations section>

BEGIN

<executable command(s)>

EXCEPTION

<exception handling goes here >

    WHEN exception1 THEN
```

```

        exception1-handling-statements

    WHEN exception2 THEN

        exception2-handling-statements

    WHEN exception3 THEN

        exception3-handling-statements

.....

    WHEN others THEN

        exception3-handling-statements

END;
```

### Example

Let us write a code to illustrate the concept. We will be using the CUSTOMERS table we had created and used in the previous chapters –

```

DECLARE

    c_id customers.id%type :=8;

    c_name customerS.Name%type;

    c_addr customers.address%type;

BEGIN

    SELECT name,address INTO c_name, c_addr

    FROM customers

    WHERE id = c_id;

    DBMS_OUTPUT.PUT_LINE ('Name: '|| c_name);

    DBMS_OUTPUT.PUT_LINE ('Address: '|| c_addr);
```

```
EXCEPTION
```

```
    WHEN no_data_found THEN
```

```
        dbms_output.put_line('No such customer!');
```

```
    WHEN others THEN
```

```
        dbms_output.put_line('Error!');
```

```
END;
```

```
/
```

When the above code is executed at the SQL prompt, it produces the following result –

```
No such customer!
```

```
PL/SQL procedure successfully completed.
```

The above program displays the name and address of a customer whose ID is given. Since there is no customer with ID value 8 in our database, the program raises the run-time exception **NO\_DATA\_FOUND**, which is captured in the **EXCEPTION block**.

### Raising Exceptions

Exceptions are raised by the database server automatically whenever there is any internal database error, but exceptions can be raised explicitly by the programmer by using the command **RAISE**. Following is the simple syntax for raising an exception –

```
DECLARE
```

```
    exception_name EXCEPTION;
```

```
BEGIN
```

```
IF condition THEN

    RAISE exception_name;

END IF;

EXCEPTION

    WHEN exception_name THEN

        statement;

END;
```

You can use the above syntax in raising the Oracle standard exception or any user-defined exception. In the next section, we will give you an example on raising a user-defined exception. You can raise the Oracle standard exceptions in a similar way.

### **User-defined Exceptions**

PL/SQL allows you to define your own exceptions according to the need of your program. A user-defined exception must be declared and then raised explicitly, using either a RAISE statement or the procedure **DBMS\_STANDARD.RAISE\_APPLICATION\_ERROR**.

The syntax for declaring an exception is –

```
DECLARE

    my-exception EXCEPTION;
```

#### **Example**

The following example illustrates the concept. This program asks for a customer ID, when the user enters an invalid ID, the exception **invalid\_id** is raised.

```
DECLARE

    c_id customers.id%type :=&cc_id;
```

```

c_name customerS.Name%type;

c_addr customers.address%type;

-- user defined exception

ex_invalid_id EXCEPTION;

BEGIN

IF c_id <=0 THEN

    RAISE ex_invalid_id;

ELSE

    SELECT name, address INTO c_name, c_addr

    FROM customers

    WHERE id = c_id;

    DBMS_OUTPUT.PUT_LINE ('Name: '|| c_name);

    DBMS_OUTPUT.PUT_LINE ('Address: '|| c_addr);

END IF;


EXCEPTION

WHEN ex_invalid_id THEN

    dbms_output.put_line('ID must be greater than zero!');

WHEN no_data_found THEN

    dbms_output.put_line('No such customer!');

WHEN others THEN

```

```
dbms_output.put_line('Error!');  
  
END;  
  
/
```

When the above code is executed at the SQL prompt, it produces the following result –

```
Enter value for cc_id: -6 (let's enter a value -6)  
old 2: c_id customers.id%type := &cc_id;  
new 2: c_id customers.id%type := -6;  
ID must be greater than zero!  
  
PL/SQL procedure successfully completed.
```

#### Pre-defined Exceptions

PL/SQL provides many pre-defined exceptions, which are executed when any database rule is violated by a program. For example, the predefined exception `NO_DATA_FOUND` is raised when a `SELECT INTO` statement returns no rows. The following table lists few of the important pre-defined exceptions –

Exception	Oracle Error	SQLCODE	Description
<code>ACCESS_INTO_NULL</code>	06530	-6530	It is raised when a null object is automatically assigned a value.
<code>CASE_NOT_FOUND</code>	06592	-6592	It is raised when none of the choices in the <code>WHEN</code> clause of a <code>CASE</code> statement is selected, and there is no <code>ELSE</code> clause.

COLLECTION_IS_NULL	06531	-6531	It is raised when a program attempts to apply collection methods other than EXISTS to an uninitialized nested table or varray, or the program attempts to assign values to the elements of an uninitialized nested table or varray.
DUP_VAL_ON_INDEX	00001	-1	It is raised when duplicate values are attempted to be stored in a column with unique index.
INVALID_CURSOR	01001	-1001	It is raised when attempts are made to make a cursor operation that is not allowed, such as closing an unopened cursor.
INVALID_NUMBER	01722	-1722	It is raised when the conversion of a character string into a number fails because the string does not represent a valid number.
LOGIN_DENIED	01017	-1017	It is raised when a program attempts to log on to the database with an invalid username or password.
NO_DATA_FOUND	01403	+100	It is raised when a SELECT INTO statement returns no rows.

NOT_LOGGED_ON	01012	-1012	It is raised when a database call is issued without being connected to the database.
PROGRAM_ERROR	06501	-6501	It is raised when PL/SQL has an internal problem.
ROWTYPE_MISMATCH	06504	-6504	It is raised when a cursor fetches value in a variable having incompatible data type.
SELF_IS_NULL	30625	-30625	It is raised when a member method is invoked, but the instance of the object type was not initialized.
STORAGE_ERROR	06500	-6500	It is raised when PL/SQL ran out of memory or memory was corrupted.
TOO_MANY_ROWS	01422	-1422	It is raised when a SELECT INTO statement returns more than one row.
VALUE_ERROR	06502	-6502	It is raised when an arithmetic, conversion, truncation, or sizeconstraint error occurs.
ZERO_DIVIDE	01476	1476	It is raised when an attempt is made to divide a number by zero.



## Lab Tasks

1. Run the examples given above
2. Create 4 user-defined exceptions