In this chapter, we will discuss Exceptions in PL/SQL. 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.No.ame%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** |
| ACCESS\_INTO\_NULL | 06530 | -6530 | It is raised when a null object is automatically assigned a value. |
| CASE\_NOT\_FOUND | 06592 | -6592 | It is raised when none of the choices in the WHEN clause of a CASE statement is selected, and there is no ELSE 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. |