TASK-13

PL/SQL Exception Handling

What is Exception

An error occurs during the program execution is called Exception in PL/SQL.

PL/SQL facilitates programmers to catch such conditions using exception block in the program and an appropriate action is taken against the error condition.

There are two type of exceptions:

* System-defined Exceptions
* User-defined Exceptions

PL/SQL Exception Handling

**Syntax for exception handling:**

Following is a general syntax for exception handling:

1. **DECLARE**
2. <declarations **section**>
3. **BEGIN**
4. <executable command(s)>
5. EXCEPTION
6. <exception handling goes here >
7. **WHEN** exception1 **THEN**
8. exception1-handling-statements
9. **WHEN** exception2  **THEN**
10. exception2-handling-statements
11. **WHEN** exception3 **THEN**
12. exception3-handling-statements
13. ........
14. **WHEN** others **THEN**
15. exception3-handling-statements
16. **END**;

Example of exception handling

Let's take a simple example to demonstrate the concept of exception handling. Here we are using the already created CUSTOMERS table.

SELECT\* FROM COUSTOMERS;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **NAME** | **AGE** | **ADDRESS** | **SALARY** |
| 1 | Ramesh | 23 | Allahabad | 20000 |
| 2 | Suresh | 22 | Kanpur | 22000 |
| 3 | Mahesh | 24 | Ghaziabad | 24000 |
| 4 | Chandan | 25 | Noida | 26000 |
| 5 | Alex | 21 | Paris | 28000 |
| 6 | Sunita | 20 | Delhi | 30000 |

1. **DECLARE**
2. c\_id customers.id%type := 8;
3. c\_name  customers.**name**%type;
4. c\_addr customers.address%type;
5. **BEGIN**
6. **SELECT**  **name**, address **INTO**  c\_name, c\_addr
7. **FROM** customers
8. **WHERE** id = c\_id;
9. DBMS\_OUTPUT.PUT\_LINE ('Name: '||  c\_name);
10. DBMS\_OUTPUT.PUT\_LINE ('Address: ' || c\_addr);
11. EXCEPTION
12. **WHEN** no\_data\_found **THEN**
13. dbms\_output.put\_line('No such customer!');
14. **WHEN** others **THEN**
15. dbms\_output.put\_line('Error!');
16. **END**;
17. /

After the execution of above code at SQL Prompt, it produces the following result:

No such customer!

PL/SQL procedure successfully completed.

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

Note: You get the result "No such customer" because the customer\_id used in the above example is 8 and there is no cutomer having id value 8 in that table.

If you use the id defined in the above table (i.e. 1 to 6), you will get a certain result. For a demo example: here, we are using the id 5.

1. **DECLARE**
2. c\_id customers.id%type := 5;
3. c\_name  customers.**name**%type;
4. c\_addr customers.address%type;
5. **BEGIN**
6. **SELECT**  **name**, address **INTO**  c\_name, c\_addr
7. **FROM** customers
8. **WHERE** id = c\_id;
9. DBMS\_OUTPUT.PUT\_LINE ('Name: '||  c\_name);
10. DBMS\_OUTPUT.PUT\_LINE ('Address: ' || c\_addr);
11. EXCEPTION
12. **WHEN** no\_data\_found **THEN**
13. dbms\_output.put\_line('No such customer!');
14. **WHEN** others **THEN**
15. dbms\_output.put\_line('Error!');
16. **END**;
17. /

After the execution of above code at SQL prompt, you will get the following result:

Name: alex

Address: paris

PL/SQL procedure successfully completed.

Raising Exceptions

In the case of any internal database error, exceptions are raised by the database server automatically. But it can also be raised explicitly by programmer by using command RAISE.

**Syntax for raising an exception:**

1. **DECLARE**
2. exception\_name EXCEPTION;
3. **BEGIN**
4. IF condition **THEN**
5. RAISE exception\_name;
6. **END** IF;
7. EXCEPTION
8. **WHEN** exception\_name **THEN**
9. statement;
10. **END**;

PL/SQL User-defined Exceptions

PL/SQL facilitates their users to define their own exceptions according to the need of the program. A user-defined exception can be raised explicitly, using either a RAISE statement or the procedure DBMS\_STANDARD.RAISE\_APPLICATION\_ERROR.

**Syntax for user define exceptions**

1. **DECLARE**
2. my-exception EXCEPTION;

PL/SQL Pre-defined Exceptions

There are many pre-defined exception in PL/SQL which are executed when any database rule is violated by the programs.

**For example:**NO\_DATA\_FOUND is a pre-defined exception which is raised when a SELECT INTO statement returns no rows.

Following is a list of some important pre-defined exceptions:

|  |  |  |  |
| --- | --- | --- | --- |
| **Exception** | **Oracle Error** | **SQL Code** | **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" clauses 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 s 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 size-constraint error occurs. |
| ZERO\_DIVIDE | 01476 | 1476 | It is raised when an attempt is made to divide a number by zero. |