ABHINAV RANJAN
RA1911003010003
CSE A1 SECTION
SRMIST , KTR

DBMS LAB 12 - EXCEPTION HANDLING IN PL/SQL

AIM :

To show the implementation of exception handling in PL/SQL and elaborate its types - system defined and user defined

THEORY :

Definition of Exception :

An exception is an error which disrupts the normal flow of program instructions. PL/SQL provides us the exception block which raises the exception thus helping the programmer to find out the fault and resolve it.

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;
```

Types of Exception :

System defined exceptions:
These exceptions are predefined in PL/SQL which get raised WHEN certain database rules are violated.
System-defined exceptions are further divided into two categories:

1. **Named system exceptions.**

They have a predefined name by the system like ACCESS_INTO_NULL, DUP_VAL_ON_INDEX, LOGIN_DENIED .

- NO_DATA_FOUND: It is raised WHEN a SELECT INTO statement returns no rows.
- TOO_MANY_ROWS:It is raised WHEN a SELECT INTO statement returns more than one row
- VALUE_ERROR:This error is raised WHEN a statement is executed that resulted in an arithmetic, numeric, string, conversion, or constraint error. This error mainly results from programmer error or invalid data input.

- ZERO_DIVIDE = raises exception WHEN dividing with zero.

2. **Unnamed system exceptions.**

Unnamed system exceptions:Oracle doesn't provide name for some system exceptions called unnamed system exceptions.These exceptions don't occur frequently.These exceptions have two parts code and an associated message. The way to handle to these exceptions is to assign name to them using Pragma EXCEPTION_INIT
Syntax:
PRAGMA EXCEPTION_INIT(exception_name, -error_number);

error_number are predefined and have a negative integer range from -20000 to -20999.

User defined exceptions:
This type of users can create their own exceptions according to the need and to raise these exceptions explicitly raise command is used.
Example:

- Divide non-negative integer x by y such that the result is greater than or equal to 1.
  From the given question we can conclude that there exist two exceptions
  - Division be zero.
  - If result is greater than or equal to 1 means y is less than or equal to x.

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.

ALGORITHM :

1. SYSTEM DEFINED EXCEPTION

- In this program , we take empno from the user as input.
- Based on empno , we search records from the emp table relating to that particular empno and display empname
- If no record is found , then a system defined exception - no_data_found is raised and appropriate message is displayed.

2. USER DEFINED EXCEPTION

- In this program , based on given empno as input , we are trying to display empname
- We define an exception which says that empno is invalid if it is lesser than 1000 or greater than 9999
- If empno is in appropriate range , output is displayed or else exception is raised

3. BOTH SYSTEM AND USER DEFINED EXCEPTION

- In this program , based on input empno we are trying to find the empname in emp table
- User defined exception is same as previous program , that range of empno should be between 1000 and 9999 else exception is raised
- System defined exception is the same as the first program . If no data is found then exception is raised

SOURCE CODE :

1. SYSTEM DEFINED EXCEPTION

```
declare
    emp_no number(10) := &empno;
```

```
    emp_name varchar2(10);
   begin
    select name into emp_name from emp where
     eid = emp_no;
     dbms_output.put_line('employee name is'||emp_name);
   exception
     when no_data_found then
        dbms_output.put_line('Not found' || emp_name);
End;
```

2. USER DEFINED EXCEPTION

```
declare
 emp_name varchar2(10);
 emp_number number(10);
 empno_out_of_range EXCEPTION;
begin
emp_number:=&empno;
IF emp_number > 9999 OR emp_number < 1000 then
                RAISE empno_out_of_range;
                ELSE
                select name INTO emp_name from emp where
eid=emp_number;
                dbms_output.put_line('Employee name is'||emp_name);
END IF;
EXCEPTION
                WHEN empno_out_of_range THEN
                dbms_output.put_line('Employee number'||emp_number||'is
out of range');
END;
```

3. BOTH SYSTEM AND USER DEFINED EXCEPTION

```
declare
 emp_name varchar2(10);
 emp_number number(10);
 empno_out_of_range EXCEPTION;
begin
```

```
 emp_number := &empno;
IF emp_number > 9999 OR emp_number<1000 THEN
        RAISE empno_out_of_range;
ELSE
        SELECT name into emp_name from emp where eid =
emp_number;
        dbms_output.put_line('Employee name is' || emp_name);
END IF;
EXCEPTION
        WHEN empno_out_of_range THEN
         dbms_output.put_line('Exception out of range');
        WHEN NO_DATA_FOUND THEN
         dbms_output.put_line('Employee not found');
END;
```

SCREENSHOTS :

1. SYSTEM DEFINED EXCEPTION

```
SQL> ed
Wrote file afiedt.buf

  1  declare
  2    emp_no number(10) := &empno;
  3    emp_name varchar2(10);
  4  begin
  5    select name into emp_name from emp where
  6    eid = emp_no;
  7    dbms_output.put_line('employee name is'||emp_name);
  8  exception
  9    when no_data_found then
 10      dbms_output.put_line('Not found' || emp_name);
 11* end;
SQL> /
Enter value for empno: 7468
old   2:   emp_no number(10) := &empno;
new   2:   emp_no number(10) := 7468;

PL/SQL procedure successfully completed.

SQL> set serveroutput on
SQL> /
Enter value for empno: 7468
old   2:   emp_no number(10) := &empno;
new   2:   emp_no number(10) := 7468;
employee name isJayesh

PL/SQL procedure successfully completed.

SQL> /
Enter value for empno: 9999
old   2:   emp_no number(10) := &empno;
new   2:   emp_no number(10) := 9999;
Not found

PL/SQL procedure successfully completed.

SQL>
```

2. USER DEFINED EXCEPTION

```
C:\Users\admin\Desktop\DBMS LAB\aws\instantclient_11_2\instantclient_11_2\sqlplus.exe
Connected.
SQL> ED
Wrote file afiedt.buf

  1  declare
  2   emp_name varchar2(10);
  3   emp_number number(10);
  4   empno_out_of_range EXCEPTION;
  5  begin
  6  emp_number:=&empno;
  7  IF emp_number > 9999 OR emp_number < 1000 then
  8     RAISE empno_out_of_range;
  9     ELSE
 10     select name INTO emp_name from emp where eid=emp_number;
 11     dbms_output.put_line('Employee name is'||emp_name);
 12  END IF;
 13  EXCEPTION
 14     WHEN empno_out_of_range THEN
 15     dbms_output.put_line('Employee number'||emp_number||'is out of range');
 16* END;
SQL> SET SERVEROUTPUT ON
SQL> /
Enter value for empno: 79000
old   6: emp_number:=&empno;
new   6: emp_number:=79000;
Employee number79000is out of range

PL/SQL procedure successfully completed.

SQL> /
Enter value for empno: 7468
old   6: emp_number:=&empno;
new   6: emp_number:=7468;
Employee name isJayesh

PL/SQL procedure successfully completed.

SQL>
```

3. BOTH SYSTEM AND USER DEFINED EXCEPTION

```
SQL> ed
Wrote file afiedt.buf

  1  declare
  2   emp_name varchar2(10);
  3   emp_number number(10);
  4   empno_out_of_range EXCEPTION;
  5  begin
  6   emp_number := &empno;
  7  IF emp_number > 9999 OR emp_number<1000 THEN
  8     RAISE empno_out_of_range;
  9  ELSE
 10     SELECT name into emp_name from emp where eid = emp_number;
 11     dbms_output.put_line('Employee name is' || emp_name);
 12  END IF;
 13  EXCEPTION
 14     WHEN empno_out_of_range THEN
 15      dbms_output.put_line('Exception out of range');
 16     WHEN NO_DATA_FOUND THEN
 17      dbms_output.put_line('Employee not found');
 18* END;
SQL> /
Enter value for empno: 7468
old   6:  emp_number := &empno;
new   6:  emp_number := 7468;
Employee name isJayesh

PL/SQL procedure successfully completed.

SQL> /
Enter value for empno: 9999
old   6:  emp_number := &empno;
new   6:  emp_number := 9999;
Employee not found

PL/SQL procedure successfully completed.

SQL>
```

RESULT :

Thus we have successfully implemented exception handling in PL/SQL