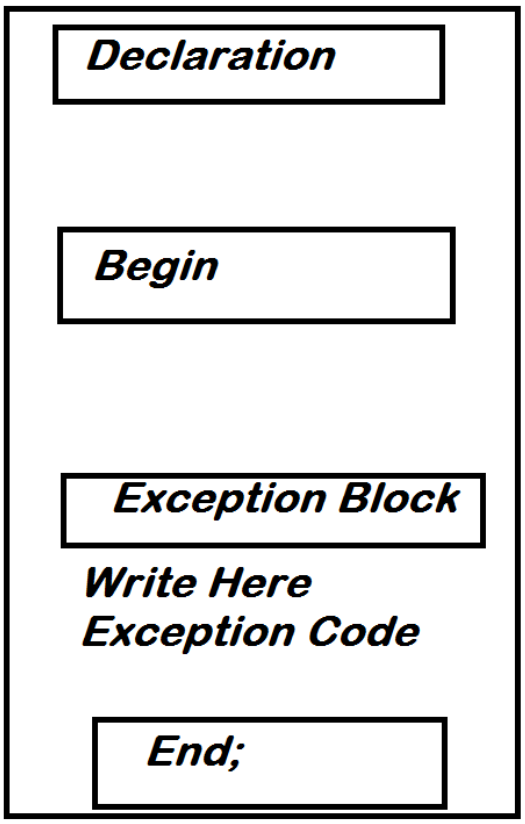
EXCEPTION HANDLING IN PLSQL

What is exception ?

Exceptions are abnormal condition that can be occur during the execution of program.

Exceptions are used to handled run-time errors in PLSQL. And to handled that exception PLSQL specified exception block is called Exception Block.

E.g.



Exceptions are two types

1. System Defined Exception
2. User Defined Exception

System Defined Exception

System Defined Exceptions are defined by Oracle Server. This exceptions contains some predefined error numbers with predefined error messages.

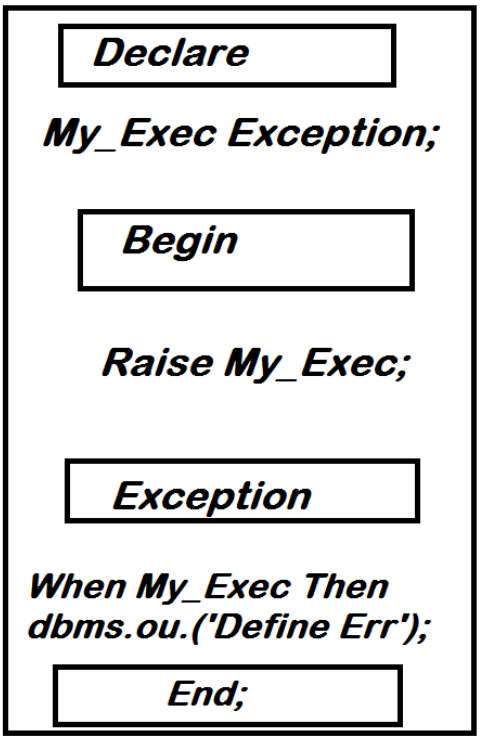
These numbers and messages are

SYSTEM_EXCEPTION	ERROR_NO	DESCRIPTION
NO_DATA_FOUND	ORA-01403	QUERY RETURN NO DATA
INVALID_CURSOR	ORA-01001	ILLEGAL CURSOR OPERATION OCCURRED IN PROGRAM
CURSOR_ALREADY_OPEN	ORA-06511	TRY TO OPEN “ALREADY OPENED CURSOR”
INVALID_NUMBER	ORA-01722	FAILED TO CONVERT FROM CHARACTER TO NUMBER
ZERO_DIVIDE	ORA-01476	ATTEMT TO DIVIDE BY ZERO (ARITHMETICEXCEPTION IN JAVA)
Prepared By JAVATECH “Search us in The World”		

User defined Exception

This exception is defined by user. You declare identifier with exception datatype. And it will de declare in declare section. And raised it inside the begin block. And if exception generated then control will automatically transferred to exception block. And exception message will displayed. There are some exception which are generated in our program but system is not defined. To control that exception we use user defined exceptions

EXCEPTION STRUCTURE



Above diagram In User Defined Exception we use Raise keyword to send exception to specified identifier.

Description

1. Declare part is used to declare variable, Exception Variable & Constants.
2. My_Exec is a variable which is defined Exception type. When exception will generate then My_Exce part will be executed. In java we write class My_Exec extends Exception to create user defined exception.
3. Between begin & Exception part your exception will be generate. So if you think exception may be occur then use raise keyword and give the exception variable name. Just I wrote Raise My_Exec.
4. Define that exception with error message. You will write

when My_Exec then
dbms_output.put_line(‘Your own error message’);
end;
Now I will first show you one by one system defined exception then I will go to user defined exception.

My_Table name STD99;

```
SQL> select * from std99;

      ROLL NAME                      MARK
---------- -------------------- ----------
         1 arun                        600
         2 kumar                       500
         3 rajesh                      400
         4 kum                         200
         6 kumi                        900
```

NO_DATA_FOUND Exception

Wap in PLSQL to display the row whose roll no is 9. Means roll 9 doesn’t exist then here NO_DATA_FOUND exception will generate.

File Edit Format View Help

```
declare
rol number:=&rol;
mar number;
begin
select mark into mar from std99 where roll=rol;
dbms_output.put_line('Mark is: '||mar);
exception
when NO_DATA_FOUND then
dbms_output.put_line('Roll no doesnot exist');
end;
```

Input 9. Roll Not exists. Execute Exception Part

exception called if no_data_found

```
SQL> @no_data.txt;
  11  /
Enter value for rol: 9
old   2: rol number:=&rol;
new   2: rol number:=9;
Roll no doesnot exist

PL/SQL procedure successfully completed.
```

INVALID_CURSOR Exception program

It is occurred in three conditions.

Actually rules of cursor is

1. Declare
2. Open
3. Fetch
4. Close

When it will be invalid_cursor (If you perform invalid cursor operation)

If fetch before opening cursor (I have done in my example. I first fetch then open cursor. It is wrong. Without open cursor how you will fetch first.)

You closed cursor before open

You fetched cursor after closing cursor.

```
declare
mar number;
cursor c1 is select mark from std99 where roll=2;
begin
fetch c1 into mar;------then should be second
open c1;-------should be first statement
dbms_output.put_line('mark is :  '||mar);
exception
when INVALID_CURSOR then
dbms_output.put_line('invalid cursor name used as
k1');
end;
```

```
SQL> @in_cur.txt
 12  /
invalid cursor name used as k1

PL/SQL procedure successfully completed.
```

CURSOR_ALREADY_OPENED Exception program.

It is generated if you try to open cursor which is previously already opened.

```
declare
mar number;
cursor c1 is select mark from std99 where roll=2;
begin
open c1; ---- FIRST TIME OPENED
fetch c1 into mar;
open c1;  ---- SECOND TIME OPENED [ EXCEPTION GENERATE ]
dbms_output.put_line('mark is :  '||mar);
exception
when CURSOR_ALREADY_OPEN then
dbms_output.put_line('CURSOR IS ALREADY OPENED BEFORE
FETCH STATEMENT');
end;
```

```
SQL> @a1_open.txt
 13  /
CURSOR IS ALREADY OPENED BEFORE FETCH STATEMENT

PL/SQL procedure successfully completed.
```

INVALID_NUMBER in java it is called NumberFormatException. It is generated when you try to insert string data in numeric column in databse. If string data is numeric type then oracle server implicit convert it. Otherwise raise INVALID_NUMBER Exception.

See Example.

File Edit Format View Help

```
begin
insert into std99 values('5','jkl','oop');
--5 can convert number roll type but oop canot convert
-- mark type. it raise invalid_number exception
exception
when INVALID_NUMBER then
dbms_output.put_line('INVALID CHARACTER CANOT CONVERT TO
NUMBER');
end;
```

```
SQL> @invalid_number.txt
  9  /
INVALID CHARACTER CANOT CONVERT TO NUMBER

PL/SQL procedure successfully completed.
```

ZERO_DIVIDE Exception (in java is called ArithmeticException)
When you try to divide any number by zero it generates ZERO_DIVIDE exception.
Watch example…..

```
declare
x number:=&x;
y number:=&y; --generate exception y input 0
z number;
begin
z:=x/y;
dbms_output.put_line(z);
exception
when zero_divide then
dbms_output.put_line('Divide by zero not possible');
end;
```

```
SQL> /
Enter value for x: 4
old   2: x number:=&x;
new   2: x number:=4;
Enter value for y: 0
old   3: y number:=&y; --generate exception y input 0
new   3: y number:=0; --generate exception y input 0
Divide by zero not possible

PL/SQL procedure successfully completed.
```

Now I will discuss user defined exception

Previous program system automatically know number is divide by zero in z:=x/y line. But user defined exception we check if y enter number is equal to zero then we explicitly generate exception by using raise keyword. And in raise keyword we specify exception identifier name.

E.g

MyExec Exception; here MyExec is a identifier which is declared Exception type. And we check y is zero or not if zero raise MyExec

If y=0 then

Raise MyExec;

And in Exception block previously we specified ZERO_DIVIDE system defined Exception. But here we use our defined as “MyExec” see how…

When ZERO_DIVIDE then

Dbms_output.put_line(‘Divide by zero is not possible’);

End

But User defined we write

When MyExec then

Dbms_output.put_line(‘Divide by zero is not possible’);

End;

See Exmple…..

```
declare
MyExec Exception;
x number:=&x;
y number:=&y; --generate exception y input 0
z number;
begin
if y=0 then
raise MyExec;  --here we explicit send err to MyExec
else
z:=x/y;
dbms_output.put_line(z);
end if;
exception
when MyExec then --After raise exception MyExec part executed
dbms_output.put_line('Divide by zero not possible');
end;
```

```
SQL> /
Enter value for x: 5
old   3: x number:=&x;
new   3: x number:=5;
Enter value for y: 0
old   4: y number:=&y; --generate exception y input 0
new   4: y number:=0; --generate exception y input 0
Divide by zero not possible

PL/SQL procedure successfully completed.
```

RAISE_APPLICATION_ERROR

Is there any technique without define exception block we can specify our user defined error message with our own error no. Yes there is technique called RAISE_APPLICATION_ERROR(erro_no,err_message)

Error_no range should be -20,000 to -20,999 [give the - minus symbol]

Message upto 2048 byte long

Where to specify it. You will write in begin section directly with errno and err message. Generally it is defined in your user defined exception. After use

raise_application_error your program is terminated there. No further executed.

See Examples….. with zero divide exception

```
declare
MyExec Exception;
x number:=&x;
y number:=&y;
z number;
begin
if y=0 then
raise_application_error(-20222,'Divide by zero not possible');
else
z:=x/y;
dbms_output.put_line(z);
end if;
dbms_output.put_line('Welcome to My Page'); --this line not
executed. line terminated after raise_application_error()
end;
```

```
SQL> @zero_divide.txt
 15  /
Enter value for x: 4
old   3: x number:=&x;
new   3: x number:=4;
Enter value for y: 0
old   4: y number:=&y;
new   4: y number:=0;
declare
*
ERROR at line 1:
ORA-20222: Divide by zero not possible
ORA-06512: at line 8
```

*************Thanking You.**************