
Lab session 14: Handling Exception in PL/SQL

Theory

What is an exception?

an exception is an identifier in PL/SQL and whenever it occurs it terminates a block and we specify an Exception handler to perform a specific action in case of that exception occurs.

Method of raising an exception?

An exception can be raised automatically for example in case of no row selected No_Data_Found exception occurs and this exception is identified by a number ORA-01403. **It is predefined exception.**

Exception can also be raised explicitly by issuing Raise statement with in the block. **It is user defined exception.**

Method of Handling exception?

- I. Trapping exception
- II. Propagating an exception

1. Trapping an exception:

Means successfully terminating a PL/SQL block after defining a proper handling of a exception in the exception block.

Theory

2. Propagating an exception:

If an exception is raised in the executable section and there is no corresponding exception defined in code then the PL/SQL block terminates with failure which is called as exception is propagated

What are exception Types?

1. Predefined Oracle server errors: (First Exception type)

One of approximately 20 errors that occur most often in PL/SQL code. There is no need to declare these exceptions in declaration block as they are already defined as an exception and they will be raised by oracle server Implicitly.

Trap a predefined Oracle Server error by referencing its standard name within the corresponding exception-handling routine. Some commonly used predefined exceptions are as follows:-

Exception Name	Oracle Server Error	Description
DUP_VAL_ON_INDEX	ORA-00001	Attempted to insert a duplicate value
INVALID_NUMBER	ORA-01722	Conversion of character string to number fails
NO_DATA_FOUND	ORA-01403	Single row SELECT returned no data
TIMEOUT_ON_RESOURCE	ORA-00051	Time-out occurred while Oracle is waiting for a resource
TOO_MANY_ROWS	ORA-01422	Single-row SELECT returned more than one row
VALUE_ERROR	ORA-06502	Arithmetic, conversion, truncation, or size-constraint error returned
ZERO_DIVIDE	ORA-01476	Attempted to divide by zero

Theory

Now lets see practical implementation of predefined exception of oracle

Example: Write a PL/SQL Program which takes employee number as input and Print Employee name as output. If no such employee exists, raise an exception NO_DATA_FOUND(predefined as we have seen now) to generate a message.

DECLARE

v_empno emp.empno%TYPE := &p_empno;

v_ename emp.ename%TYPE;

DEMO OF Predefined oracle exception

BEGIN

SELECT ename into v_ename

FROM EMP

WHERE empno = v_empno;

DBMS_OUTPUT.PUT_LINE ('Employee Name : ' || v_ename);

EXCEPTION

WHEN NO_DATA_FOUND THEN

DBMS_OUTPUT.PUT_LINE ('No employee with number ' || v_empno || '
Found');

END;

Theory

2. Non Predefined Oracle server errors: (Second Exception Type)

Exception is not defined but actions as a error is defined in oracle that may cause a block to terminate abnormally after informing the error so in order to not terminate a block abnormally we need to write handlers.

For example for a column on which NOT NULL constraint has been defined if a user want to insert a Null Value than oracle will show an error with a code (action of a error is define) and may terminate a block abnormally so we will write a exception to avoid this abnormal termination

For Empno column we have Not Null constraint defined so if I will write the following code then this will generate a error:

DECLARE

BEGIN

```
Insert into EMP(EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO)
values (NULL,'Rabia','Artist',7811,to_timestamp('26-JUL-81','DD-MON-RR
HH.MI.SSXFF AM'),9000,null,30);
```

EXCEPTION

```
WHEN DUP_VAL_ON_INDEX THEN
```

```
DBMS_OUTPUT.PUT_LINE ('No duplicate value is allowed');
```

END;

The above PL/SQL will terminate with a Not Null Error

Theory

The screenshot displays the SQL Developer environment. The main window shows a PL/SQL script in the 'Query Builder' tab. The script is as follows:

```
DECLARE
BEGIN
  Insert into EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO) values (NULL, 'Rabia', 'Artist', 7811, to_timestamp('26-JUL-81', 'DD-MON-RR
EXCEPTION
  WHEN DUP_VAL_ON_INDEX THEN
    DBMS_OUTPUT.PUT_LINE ('No duplicate value is allowed');
END;
```

A red arrow points from the 'Run' button (a green play icon) in the toolbar to the text 'Run Code'.

Below the script, the 'Script Output' window is open, showing the 'Error report -'. The error message is:

```
ORA-01400: cannot insert NULL into ("SCOTT"."EMP"."EMPNO")
ORA-06512: at line 3
01400. 00000 - "cannot insert NULL into (%s)"
*Cause:      An attempt was made to insert NULL into previously listed objects.
```

A red arrow points from the error message to the text 'Error'.

Below the error report, the 'Messages - Log' window is open, showing the following messages:

```
ORA-06512: at line 1

One error saving changes to table "SCOTT"."EMP"
Row 15: ORA-01400: cannot insert NULL into ("SCOTT"."EMP"."EMPNO")
ORA-06512: at line 1
```

A red arrow points from the error message in the log to the text 'Oracle 01400 predefined error'.

The bottom right corner of the slide features the text 'Lab 14' in a large, bold, blue font.

Theory

We have seen for NOT NULL error there is no predefined Exception. So we will declare this exception that has a defined error in oracle like we saw (**Oracle 01400**)

So we need to declare Non Predefined exception with predefined Oracle Error using a **PRAGMA** Keyword so our whole PL/SQL Code Will be as Following:

DECLARE

```
my_excep EXCEPTION;  
PRAGMA EXCEPTION_INIT (my_excep, -1400);
```



I have defined my exception here

BEGIN

```
Insert into EMP(EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values  
(NULL,'Rabia','Artist',7811,to_timestamp('26-JUL-81','DD-MON-RR HH.MI.SSXFF  
AM'),9000,null,30);
```

EXCEPTION

```
WHEN DUP_VAL_ON_INDEX THEN
```

```
    DBMS_OUTPUT.PUT_LINE ('No duplicate value is allowed');
```

```
WHEN my_excep THEN
```

```
    DBMS_OUTPUT.PUT_LINE ('Can not insert Null Values');
```



**What to do when
My exception occurs**

END;

Now lets update our code as above and re run it to see what output now

Theory

The screenshot displays the Oracle SQL Developer environment. The top toolbar includes a 'Run' button (a green play icon) which is highlighted by a red arrow pointing to the text 'Run Code'. Below the toolbar, the 'Worksheet' tab is active, showing a PL/SQL script. The script defines an exception 'my_excep' and attempts to insert a record with a NULL value into the 'EMP' table. The script is as follows:

```
DECLARE
my_excep EXCEPTION ;
PRAGMA EXCEPTION_INIT (my_excep, -1400);
BEGIN
  Insert into EMP(EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO) values (NULL,'Rabia','Artist',7811,to_timestamp('26-JUL-81','DD-MON-RR
EXCEPTION
  WHEN DUP_VAL_ON_INDEX THEN
    DBMS_OUTPUT.PUT_LINE ('No duplicate value is allowed');
  WHEN my_excep THEN
    DBMS_OUTPUT.PUT_LINE ('Can not insert Null Values');
END;
```

The 'Script Output' window at the bottom shows the results of the execution. It contains the following text:

```
PL/SQL procedure successfully completed.

Can not insert Null Values

PL/SQL procedure successfully completed.
```

A red arrow points from the 'Can not insert Null Values' message in the output window back to the corresponding line in the script.

Theory

3. User Defined Exception (Third Exception Type)

Any normal condition that the developer determines is abnormal. We Declare this within the declarative section and raise explicitly.

For example if I say that salary greater than 3000 is a exception. So it clearly shows this is a normal executable condition but user want to define this as a exception via code as follows

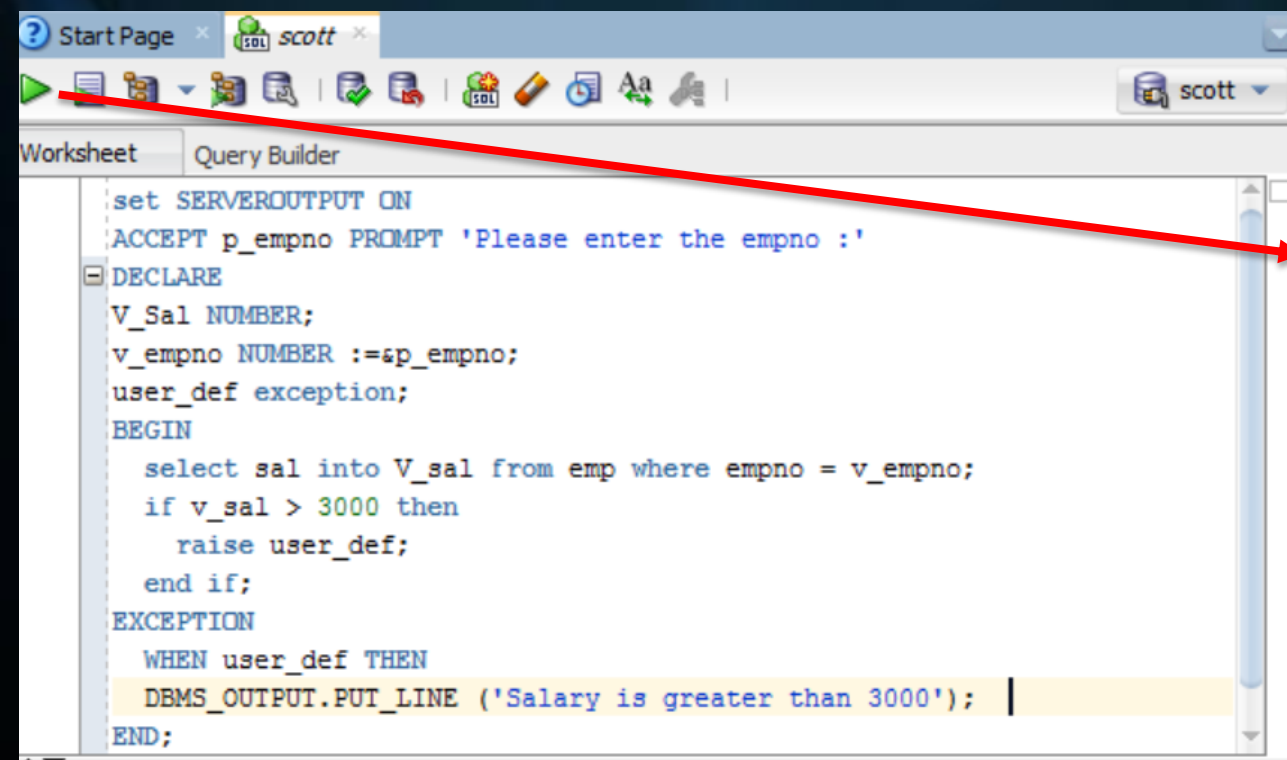
```
set SERVEROUTPUT ON
ACCEPT p_empno PROMPT 'Please enter the empno : '
DECLARE
    V_Sal NUMBER;
    v_empno NUMBER := &p_empno;
    user_def exception;
BEGIN
    select sal into V_sal from emp where empno = v_empno;
    if v_sal > 3000 then
        raise user_def;
    end if;
EXCEPTION
    WHEN user_def THEN
        DBMS_OUTPUT.PUT_LINE ('Salary is greater than 3000');
END;
```

Condition that will cause a exception

Rasing exception

Action when exception occurs

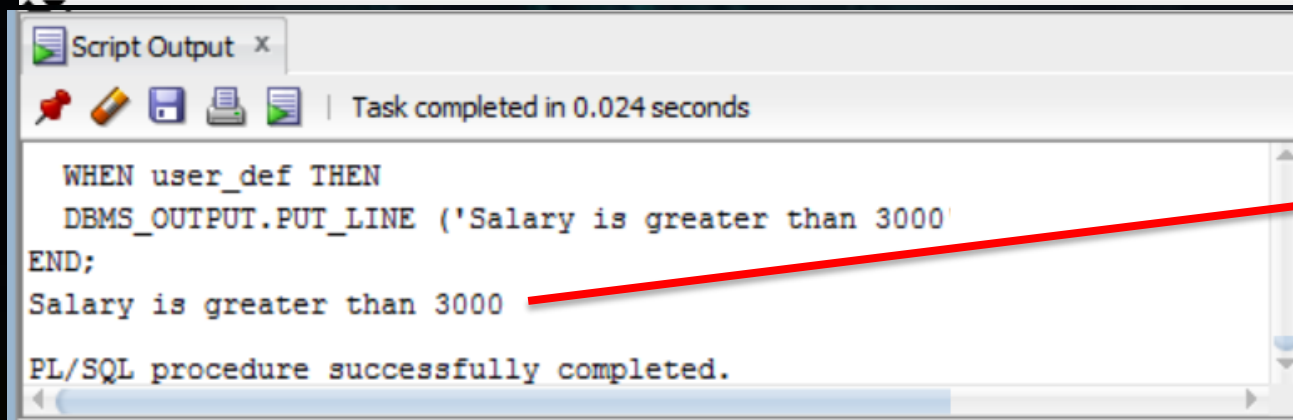
Theory



```
set SERVEROUTPUT ON
ACCEPT p_empno PROMPT 'Please enter the empno : '
DECLARE
V_sal NUMBER;
v_empno NUMBER := &p_empno;
user_def exception;
BEGIN
  select sal into V_sal from emp where empno = v_empno;
  if v_sal > 3000 then
    raise user_def;
  end if;
EXCEPTION
  WHEN user_def THEN
    DBMS_OUTPUT.PUT_LINE ('Salary is greater than 3000');
END;
```

Run Code

See user defined exception
DEMO



Script Output x

Task completed in 0.024 seconds

```
WHEN user_def THEN
  DBMS_OUTPUT.PUT_LINE ('Salary is greater than 3000')
END;
Salary is greater than 3000
PL/SQL procedure successfully completed.
```

I have placed King Emp id which
Have 5000 sal.