Exceptions are runtime errors or unexpected events that occur during the execution of a PL/SQL code block.Whenever runtime error occur use an appropriate exception name in exception handler under exception section. An error which is handled by DBA or technical user. *Using exception we can display user-friendly messages.*

Exception Handling

•PL/SQL provides a feature to handle the Exceptions which occur in a PL/SQL Block known as exception Handling.
•Using Exception Handling we can test the code and avoid it from exiting shortly.
• When an exception occurs a messages which explains its cause is received.
•Exception handling can be done in the EXCEPTION part of PL/SQL program code block
Syntax for exception handling

```
DECLARE
  Declaration section
BEGIN
  Exception section
EXCEPTION
WHEN ex_name1 THEN
   -Error handling statements
WHEN ex_name2 THEN
   -Error handling statements
WHEN Others THEN
  -Error handling statements
END;
```

•When an exception is raised, Oracle searches for an appropriate exception handler in the exception section.
•For example in the above example, if the error raised is 'ex_name1 ', then the error is handled according to the statements under it.
•Since, it is not possible to determine all the possible runtime errors during testing fo the code, the 'WHEN Others' exception is used to manage the exceptions that are not explicitly handled.
• Only one exception can be raised in a Block and the control does not return to the Execution Section after
the error is handled.

There are 3 types of Exceptions.

a) Named System Exceptions
b) Unnamed System Exceptions
c) User-defined Exceptions

Named System Exceptions

• System exceptions are automatically raised by Oracle, when a program violates a RDBMS rule.
• There are some system exceptions which are raised frequently, so they are pre-defined and given a name in
Oracle which are known as Named System Exceptions.
For example:
NO_DATA_FOUND and ZERO_DIVIDE are called Named System exceptions.
Named system exceptions are:
1) Not Declared explicitly,

2) Raised implicitly when a predefined Oracle error occurs,
3) caught by referencing the standard name within an exception-handling routine

Some named system exception in oracle

Oracle Exception Name	Oracle Error	Explanation
DUP_VAL_ON_INDEX	ORA-00001	You tried to execute an INSERT or UPDATE statement that has created a duplicate value in a field restricted by a unique index.
TIMEOUT_ON_RESOURCE	ORA-00051	You were waiting for a resource and you timed ou
TRANSACTION_BACKED_OUT	ORA-00061	The remote portion of a transaction has rolled back.
INVALID_CURSOR	ORA-01001	You tried to reference a cursor that does not yet exist. This may have happened because you've executed a FETCH cursor or CLOSE cursor before OPENing the cursor.
NOT_LOGGED_ON	ORA-01012	You tried to execute a call to Oracle before logging in.
LOGIN_DENIED	ORA-01017	You tried to log into Oracle with an invalid username/password combination.
NO_DATA_FOUND	ORA-01403	You tried one of the following: 1. You executed a SELECT INTO statement an no rows were returned. 2. You referenced an uninitialized row in a table. 3. You read past the end of file with the UTL_FILE package.
TOO_MANY_ROWS	ORA-01422	You tried to execute a SELECT INTO statement an more than one row was returned.
ZERO_DIVIDE	ORA-01476	You tried to divide a number by zero.
INVALID_NUMBER	ORA-01722	You tried to execute a SQL statement that tried to convert a string to a number, but it was unsuccessful.
STORAGE_ERROR	ORA-06500	You ran out of memory or memory was corrupted
PROGRAM_ERROR	ORA-06501	This is a generic "Contact Oracle support" messag because an internal problem was encountered.
VALUE_ERROR	ORA-06502	You tried to perform an operation and there was a error

Oracle Exception Name	Oracle Error	Explanation
		on a conversion, truncation, or invalid constrainin of numeric or character data.
CURSOR_ALREADY_OPEN	ORA-06511	You tried to open a cursor that is already open.

Some Predefined Exception:

1) no_data_found
2) too_many_rows
3) zero_divide
4) invalid_cursor
5) cursor_already_open
6) invalid_number
7) value_error

1) no_data_found exception:

When a pl/sql block contains select _____ into clause and also if requested data not available in a table oracle server return an error *ora-1403: no data found*. To handle this error we are using no_data_found exception name.

Example: set serveroutput on

```
declare
vename varchar2(10);
vsal number(10);
begin
select ename,sal into vename,vsal from emp where eno=&no;
dbms_output.put_line(vename || '    ' || vsal);
exception
when no_data_found then
dbms_output.put_line('Emp number does not exists');
end;
/
```

2) too_many_rows exception

When a pl/sql block contains select _____ into clause try to return more than one record or value then oracle server return an error *ora-1422: exact fetch returns more than requested number of rows.* To handle this error we are using too_many_rows exception name.

Example: set serveroutput on

```
declare
vsal number(10);
begin
select sal into vsal from emp;
dbms_output.put_line(vsal);
exception
when too_many_rows then
dbms_output.put_line('not return more rows or values');
end;
/
```

3) zero_divide exception

Ora-1476: Divisor is equal to zero.

Example: set serveroutput on

```
declare
a number(2);
b number(2);
c number(2);
begin
a:=&a;
b:=&b;
c:=a/b;
dbms_output.put_line(c);
exception
when zero_divide then
dbms_output.put_line('b can't be declare zero');
end;
/
```

4) invalid_cursor exception

Whenever we are performing invalid operations on the cursor oracle server returns an error i.e. if you are try to close the cursor without opening the cursor, oracle server return an error *ora-1001: invalid cursor.* To handle this error we are using invalid_cursor exception name.

Example: set serveroutput on

```
declare
cursor c1 is select * from emp;
I emp%rowtype;
Begin
Loop
Fetch c1 into I;
Exit when c1%notfound;
dbms_output.put_line(i.ename||'    '||i.sal);
end loop;
close c1;
exception
when invalid_cursor then
dbms_output.put_line('First we must open the cursor');
end;
/
```

5) cursor_already_cursor exception

When we are try to reopen the cursor without closing the cursor, oracle server return an error *ora-6511: cursor already open.* To handle this error we are using cursor_already_open exception name.

Example: set serveroutput on

```
declare
cursor c1 is select * from emp;
I emp%rowtype;
Begin
Open c1;
Loop
Fetch c1 into I;
Exit when c1%notfound;
dbms_output.put_line(i.ename||'    '||i.sal);
end loop;
```

```
open c1;
exception
when cursor_already_open then
dbms_output.put_line('Before reopen we must close the cursor');
end;
/
```

Whenever we are try to convert string type to number type oracle server return errors invalid number, value error.

6) invalid_number exception

When a pl/sql blocks contains sql statements and also if you are tries to convert string type to number type oracle server returns an errors *ora-1722: invalid number.* To handle this error we are using invalid number exception name.

Example: set serveroutput on

```
Begin
Insert into emp(eno,sal) values(1,'abc');
exception
when invalid_number then
dbms_output.put_line('Insert proper Data only');
end;
/
```

7) value_error exception

When a pl/sql blocks contains pl/sql statements and also if you are tries to convert string type to number type oracle server returns an errors *ora-6502: numeric or value error: character to number conversion error.* To handle this error we are using value_error exception name.

Example: set serveroutput on

```
declare
z number(10);
begin
z:= '&x'+'&y';
dbms_output.put_line(z);
exception
when value_error then
dbms_output.put_line('Enter proper data only ');
end;
/
```

Whenever we are try to store large amount of data than the specified datatype size in variable declaration oracle server return an error ora:6502: numeric or value error: character string buffer too small. To handle this error we are using value_error exception name.

Example: set serveroutput on

```
declare
z varchar(3);
begin
z:= 'abcd';
dbms_output.put_line(z);
exception
when value_error then
dbms_output.put_line('Invalid String length');
end;
/
```

Unnamed System Exceptions

•Those system exception for which oracle does not provide a name is known as unamed system exception.
•These exception do not occur frequently.
•These Exceptions have a code and an associated message.
•There are two ways to handle unnamed sysyem exceptions:
1. By using the WHEN OTHERS exception handler, or
2. By giving a name to the exception code and using it as a named exception. We can assign a name to unnamed system exceptions using a Pragma called EXCEPTION_INIT.

```
declare
 r number;
 a std%rowtype;
begin
   select * into a  from std where rollno=&r;
   dbms_output.put_line('Name -' || a.name || '  ' || 'Address -' || '  '  ||a.address);
Exception
   when too_many_rows then
    dbms_output.put_line('There are multiple rows related to rollno');
 when others then
     dbms_output.put_line('There is exception which is unknown');
end;
```

pragma EXCEPTION_INIT: Pragma is a keyword directive to execute proceed at compile time.
pragma EXCEPTION_INIT function take this two argument,

1) exception_name
2) error_number

```
DECLARE
 exp exception;
pragma exception_init (exp,-20015);
n int:=10;
BEGIN
 FOR i IN 1..n LOOP
dbms_output.put_line(i*i);
IF i=6 THEN
 RAISE exp;
END IF;
END LOOP;
EXCEPTION
WHEN exp THEN
dbms_output.put_line('6 round completed ');
END;
```

User-defined Exceptions

we can explicitly define exceptions based on business rules. These are known as user-defined exceptions.
Steps to be followed to use user-defined exceptions:

1. They should be explicitly declared in the declaration section.
2. They should be explicitly raised in the Execution Section.
3. They should be handled by referencing the user-defined exception name in the exception

section.

```
declare
  checkage Exception;
   a number;
begin
   a:=&a;
   if(a<18) then
    raise checkage;
   end if;
    Dbms_output.put_line('Your are eligiable for vote');
    Exception
     when checkage then
      dbms_output.put_line('User Defined exception for bellow 18 years age');
end;
/
```

RAISE_APPLICATION_ERROR

- RAISE_APPLICATION_ERROR is a built-in procedure in oracle which is used to display the user-defined error messages along with the error number whose range is in between -20000 and -20999.
- Whenever a message is displayed using RAISE_APPLICATION_ERROR, all previous transactions which are not committed within the PL/SQL Block are rolled back automatically (i.e. change due to INSERT, UPDATE, or DELETE statements).
- RAISE_APPLICATION_ERROR raises an exception but does not handle it.
- RAISE_APPLICATION_ERROR is used for the following reasons,
  a) to create a unique id for an user-defined exception.
  b) to make the user-defined exception look like an Oracle error.
- The General Syntax to use this procedure is:

RAISE_APPLICATION_ERROR (error_number, error_message);

The Error number must be between -20000 and -20999

The Error_message is the message you want to display when the error occurs.

Steps to be followed to use RAISE_APPLICATION_ERROR procedure:

1. Declare a user-defined exception in the declaration section.
2. Raise the user-defined exception based on a specific business rule in the execution section.
3. Finally, catch the exception and link the exception to a user-defined error number in RAISE_APPLICATION_ERROR.

```
declare
  checkage Exception;
   a number;
begin
   a:=&a;
   if(a<18) then
    raise checkage;
   end if;
    Dbms_output.put_line('Your are eligiable for vote');
    Exception
     when checkage then
      raise_application_error(-20004,'User Defined exception for bellow 18 years age');
end;
```