Error Handling

Exception

- An exception is an error condition during a program execution.
- PL/SQL supports programmers to catch such conditions using **EXCEPTION** block in the program and an appropriate action is taken against the error condition.
- There are two types of exceptions −
  - System-defined exceptions
  - User-defined exceptions / Predefined exceptions

Continued

- Exceptions are designed for run time error handling, rather than compile time error handling.

Error type	**Reported by**	**How handled**
Compile – time	PL/SQL compiler	Interactively – compiler reports errors, and you have to correct them.
Runtime	PL/SQL runtime engine	Programmatically – exceptions are raised and caught by exception handlers.

Syntax

```
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;
```

Example – divide by zero

```
> DECLARE v_invalid INTEGER;
> BEGIN
>         v_invalid := 100/0;
> EXCEPTION
>         WHEN ZERO_DIVIDE THEN
>          DBMS_OUTPUT.PUT_LINE ('Attempt to divide by 0');
> END;
> /
Attempt to divide by 0

PL/SQL procedure successfully completed.
```

Example – Customer table

```
DECLARE
   c_id customers.id%type := 8;
   c_name customerS.Name%type;
   c_addr customers.address%type;
BEGIN
   SELECT  name, address INTO
c_name,       c_addr  FROM
customers    WHERE id = c_id;
   DBMS_OUTPUT.PUT_LINE ('Name:
'||      c_name);
   DBMS_OUTPUT.PUT_LINE
('Address: ' ||          c_addr);
EXCEPTION
   WHEN no_data_found THEN
      dbms_output.put_line('No such
         customer!');
   WHEN others THEN
      dbms_output.put_line('Error!');
END;
```

- When the above code is executed at the SQL prompt, it produces the following result −

```
No such customer!
PL/SQL procedure successfully completed.
```

System defined exception/ Predefined exception

Exception	Description
ACCESS_INTO_NULL	It is raised when a null object is automatically assigned a value.
CASE_NOT_FOUND	It is raised when none of the choices in the WHEN clause of a CASE statement is selected, and there is no ELSE clause.
DUP_VAL_ON_INDEX	It is raised when duplicate values are attempted to be stored in a column with unique index.
INVALID_CURSOR	It is raised when attempts are made to make a cursor operation that is not allowed, such as closing an unopened cursor.
INVALID_NUMBER	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	It is raised when a program attempts to log on to the database with an invalid username or password.

Continued

ROWTYPE_MISMATCH	It is raised when a cursor fetches value in a variable having incompatible data type.
STORAGE_ERROR	It is raised when PL/SQL ran out of memory or memory was corrupted.
TOO_MANY_ROWS	It is raised when a SELECT INTO statement returns more than one row.
VALUE_ERROR	It is raised when an arithmetic, conversion, truncation, or sizeconstraint error occurs.
ZERO_DIVIDE	It is raised when an attempt is made to divide a number by zero.

User defined exception

PL/SQL allows you to define your own exceptions according to the need of your program.

A user-defined exception must be declared and raised.

Syntax :

```
DECLARE
   my-exception EXCEPTION;
```

Example (table – customer)

```
DECLARE
   c_id customers.id%type := &cc_id;
   c_name customers.Name%type;
   c_addr customers.address%type;
   -- user defined exception
   ex_invalid_id  EXCEPTION;
BEGIN
   IF c_id <= 0 THEN
      RAISE ex_invalid_id;
   ELSE
      SELECT  name, address INTO  c_name, c_addr
      FROM customers
      WHERE id = c_id;
DBMS_OUTPUT.PUT_LINE ('Name: '|| c_name);
      DBMS_OUTPUT.PUT_LINE ('Address: ' || c_addr);
   END IF;
EXCEPTION
   WHEN ex_invalid_id THEN
      dbms_output.put_line('ID must be greater than      zero!');
   WHEN no_data_found THEN
      dbms_output.put_line('No such customer!');
   WHEN others THEN
      dbms_output.put_line('Error!');
END;
```