Oracle (PL/SQL)

Lesson 3: Exception Handling

Lesson Objectives

- To understand the following topics:
  - Error Handling
    - Predefined Exceptions
    - Numbered Exceptions
    - User Defined Exceptions
  - Raising Exceptions
  - Control passing to Exception Handlers
  - RAISE_APPLICATION_ERROR

Understanding Exception Handling in PL/SQL

- Error Handling:
  - In PL/SQL, a warning or error condition is called an “exception”.
    - Exceptions can be internally defined (by the run-time system) or user defined.
    - Examples of internally defined exceptions:
  - division by zero
  - out of memory
    - Some common internal exceptions have predefined names, namely:
  - ZERO_DIVIDE
  - STORAGE_ERROR

Understanding Exception Handling in PL/SQL

- The other exceptions can be given user-defined names.
- Exceptions can be defined in the declarative part of any PL/SQL block, subprogram, or package. These are user-defined exceptions.

Declaring Exception

- Exception is an error that is defined by the program.
  - It could be an error with the data, as well.
- There are three types of exceptions in Oracle:
  - Predefined exceptions
  - Numbered exceptions
  - User defined exceptions

Predefined Exception

- Predefined Exceptions correspond to the most common Oracle errors.
  - They are always available to the program. Hence there is no need to declare them.
  - They are automatically raised by ORACLE whenever that particular error condition occurs.
  - Examples: NO_DATA_FOUND,CURSOR_ALREADY_OPEN, PROGRAM_ERROR

Predefined Exception - Example

- In the following example, the built in exception is handled.

```
DECLARE
        v_staffno   staff_master.staff_code%type;
        v_name      staff_master.staff_name%type;
BEGIN
        SELECT staff_name into v_name FROM staff_master
        WHERE staff_code=&v_staffno;
        dbms_output.put_line(v_name);
EXCEPTION
        WHEN  NO_DATA_FOUND THEN
        dbms_output.put_line('Not Found');
END;
 /
```

Numbered Exception

- An exception name can be associated with an ORACLE error.
  - This gives us the ability to trap the error specifically to ORACLE errors
  - This is done with the help of “compiler directives” –
  - PRAGMA EXCEPTION_INIT

Numbered Exception

- PRAGMA EXCEPTION_INIT:
  - A PRAGMA is a compiler directive that is processed at compile time, not at run time. It is used to name an exception.
  - In PL/SQL, the PRAGMA EXCEPTION_INIT tells the compiler to associate an exception name with an Oracle error number.
    - This arrangement lets you refer to any internal exception(error) by name, and to write a specific handler for it.
  - When you see an error stack, or sequence of error messages, the one on top is the one that you can trap and handle.

Numbered Exception (Contd.)

- User defined exceptions can be named with error number between -20000 and -20999.
- The naming is declared in Declaration section.
- It is valid within the PL/SQL blocks only.
- Syntax is:

```
PRAGMA EXCEPTION_INIT(Exception Name,Error_Number);
```

Numbered Exception - Example

- A PL/SQL block to handle Numbered Exceptions

```
DECLARE
        v_bookno number := 10000008;
       child_rec_found EXCEPTION;
       PRAGMA EXCEPTION_INIT (child_rec_found, -2292);
BEGIN
            DELETE from book_master
          WHERE book_code = v_bookno;
EXCEPTION
            WHEN child_rec_found THEN
      INSERT into error_log
            VALUES (‘Book entries exist for book:' || v_bookno);
END;
```

User-defined Exception

- User-defined Exceptions are:
  - declared in the Declaration section,
  - raised in the Executable section, and
  - handled in the Exception section

User-defined Exception - Example

- Here is an example of User Defined Exception:

```
DECLARE
    E_Balance_Not_Sufficient EXCEPTION;
    E_Comm_Too_Large EXCEPTION;
    …
BEGIN
    NULL;
END;
```

Raising Exceptions

- Raising Exceptions:
  - Internal exceptions are raised implicitly by the run-time system, as are user-defined exceptions that are associated with an Oracle error number using EXCEPTION_INIT.
  - Other user-defined exceptions must be raised explicitly by RAISE statements.
    - The syntax is:

```
RAISE  Exception_Name;
```

Raising Exceptions - Example

- An exception is defined and raised as shown below:

```
DECLARE
        ...
        retired_emp EXCEPTION ;
BEGIN
        pl/sql_statements ;
        if error_condition then
        RAISE retired_emp ;
        pl/sql_statements ;
EXCEPTION
        WHEN retired_emp THEN
        pl/sql_statements ;
END ;
```

Control passing to Exception Handler

- Control passing to Exception Handler :
  - When an exception is raised, normal execution of your PL/SQL block or subprogram stops, and control passes to its exception-handling part.
  - To catch the raised exceptions, you write “exception handlers”.
    - Each exception handler consists of a WHEN clause, which specifies an exception, followed by a sequence of statements to be executed when that exception is raised.
  - These statements complete execution of the block or subprogram, however, the control does not return to where the exception was raised. In other words, you cannot resume processing where you left off.

User-defined Exception - Example

- User Defined Exception Handling:

```
DECLARE
      dup_deptno EXCEPTION;
      v_counter binary_integer;
      v_department number(2) := 50;
BEGIN
     SELECT count(*) into v_counter  FROM department_master
      WHERE dept_code=50;
  IF v_counter > 0 THEN
            RAISE dup_deptno ;
  END IF;
      INSERT into department_master
      VALUES (v_department ,'new name');
EXCEPTION
       WHEN dup_deptno THEN
         INSERT into error_log
         VALUES (‘Dept: ‘|| v_department ||' already exists”);
   END ;
```

Others Exception Handler

- OTHERS Exception Handler:
  - The optional OTHERS exception handler, which is always the last handler in a block or subprogram, acts as the handler for all exceptions that are not specifically named in the Exception section.
  - A block or subprogram can have only one OTHERS handler.
  - To handle a specific case within the OTHERS handler, predefined functions SQLCODE and SQLERRM are used.
    - SQLCODE returns the current error code. And SQLERRM returns the current error message text.
    - The values of SQLCODE and SQLERRM should be assigned to local variables before using it within a SQL statement.

Others Exception Handler - Example

```
DECLARE
  v_dummy varchar2(1);
  v_designation number(2) := 109;
BEGIN
      SELECT 'x' into v_dummy FROM designation_master
      WHERE design_code= v_designation;
       INSERT into error_log
      VALUES (‘Designation: ' || v_designation || 'already exists');
EXCEPTION
     WHEN no_data_found THEN
          insert into designation_master  values (v_designation,’newdesig’);
      WHEN OTHERS THEN
           Err_Num = SQLCODE;
           Err_Msg =SUBSTR( SQLERRM, 1, 100);
            INSERT into errors  VALUES( err_num, err_msg );
END ;
```

Raise_Application_Error

- RAISE_APPLICATION_ERROR:
  - The procedure RAISE_APPLICATION_ERROR lets you issue user-defined ORA-error messages from stored subprograms.
  - In this way, you can report errors to your application and avoid returning unhandled exceptions.
  - Syntax:

```
RAISE_APPLICATION_ERROR( Error_Number, Error_Message);
```

  - where:
    - Error_Number is a parameter between -20000 and -20999
    - Error_Message is the text associated with this error

Raise_Application_Error - Example

- Here is an example of Raise Application Error:

```
DECLARE
    /* VARIABLES */
BEGIN
    ……..
    ……..
EXCEPTION
    WHEN OTHERS THEN
    -- Will transfer the error to the calling environment
    RAISE_APPLICATION_ERROR( -20999 ,'Contact DBA');
END ;
```

Summary

- In this lesson, you have learnt about:
  - Exception Handling
    - User-defined Exceptions
    - Predefined Exceptions
  - Control passing to Exception Handler
  - OTHERS exception handler
  - Association of Exception name to Oracle errors
  - RAISE_APPLICATION_ERROR procedure

Review Question

- Question 1: The procedure ___ lets you issue user-defined ORA-error messages from stored subprograms
- Question 2: The ___ tells the compiler to associate an exception name with an Oracle error number.
- Question 3: ___ returns the current error code. And ___ returns the current error message text.