



Lesson Objectives



To understand the following topics:

- Error Handling
- Declaring Exceptions
 - Predefined Exceptions
 - User Defined Exceptions
 - Raising Exceptions
- OTHERS exception handler

2.1: Error Handling (Exception Handling)



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
 - 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.

Error Handling:

- A good programming language should provide capabilities of handling errors and recovering from them if possible.
- PL/SQL implements Error Handling via “exceptions” and “exception handlers”.

Types of Errors in PL/SQL

- **Compile Time errors:** They are reported by the PL/SQL compiler, and you have to correct them before recompiling.
- **Run Time errors:** They are reported by the run-time engine. They are handled programmatically by raising an exception, and catching it in the Exception section.

2.1: Error Handling (Exception Handling)

**Declaring Exception**

Exception is an error that is defined by the program.

- It could be an error with the data, as well.

There are two types of exceptions in Oracle:

- Predefined exceptions
- User defined exceptions

Declaring Exceptions:

- Exceptions are declared in the Declaration section, raised in the Executable section, and handled in the Exception section.

2.2: Predefined 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 Exceptions:

- An internal exception is raised implicitly whenever your PL/SQL program violates an Oracle rule or exceeds a system-dependent limit. Every Oracle error has a number, but exceptions must be handled by name. So, PL/SQL predefines some common Oracle errors as exceptions. For example, PL/SQL raises the predefined exception NO_DATA_FOUND if a SELECT INTO statement returns no rows.
- Given below are some Predefined Exceptions:
 - NO_DATA_FOUND
 - This exception is raised when SELECT INTO statement does not return any rows.
 - TOO_MANY_ROWS
 - This exception is raised when SELECT INTO statement returns more than one row.
 - INVALID_CURSOR
 - This exception is raised when an illegal cursor operation is performed such as closing an already closed cursor.
 - VALUE_ERROR
 - This exception is raised when an arithmetic, conversion, truncation, or constraint error occurs in a procedural statement.
 - DUP_VAL_ON_INDEX
 - This exception is raised when the UNIQUE CONSTRAINT is violated.

2.2: Predefined Exceptions



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;
/
```

Predefined Exceptions:

In the example shown on the slide, the NO_DATA_FOUND built in exception is handled. It is automatically raised if the SELECT statement does not fetch any value and populate the variable.

2.3: User defined Exceptions



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 Exceptions:

- These exception are entirely user defined based on the application. The programmer is responsible for declaring, raising and handling them.

2.3: User defined Exceptions



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;
```


2.3: User defined Exceptions



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:

When the error associated with an exception occurs, the exception is raised. This is done through the RAISE command.

2.3: User defined Exceptions



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 ;
```

2.3: User defined Exceptions

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 ;
```

The example on the slide demonstrates user-defined exceptions. It checks for department no value to be inserted in the table. If the value is duplicated it will raise an exception.

2.4: OTHERS Exception Handler



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.

2.4: OTHERS Exception Handler



OTHERS Exception Handler (contd..)

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.

2.4: OTHERS Exception Handler

OTHERS Exception Handler - Example

```
DECLARE
  v_dummy varchar2(1);
  v_designation number(3) := 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 ;
```

The example on the slide uses OTHERS Exception handler. If the exception that is raised by the code is not NO_DATA_FOUND, then it will go to the OTHERS exception handler since it will notice that there is no appropriate exception handler defined.

Also observe that the values of SQLCODE and SQLERRM are assigned to variables defined in the block.

Summary

In this lesson, you have learnt about:

- Exception Handling
 - Predefined Exceptions
 - User-defined Exceptions
- OTHERS exception handler



Answers for Review Questions:

Question 1:
Answer: SQLERRM

Question 2:
Answer: SQLCODE

Review – Questions

Question 1: ____ returns the current error message text.

Question 2: ____ returns the current error code.

