Oracle for Developers (PL/SQL)

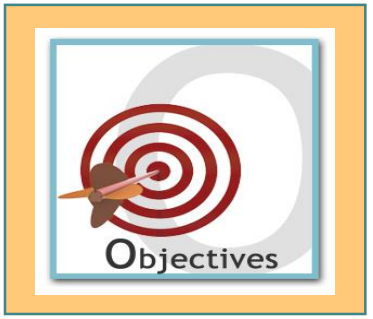
Built-in Packages in Oracle

Capgemini

Lesson Objectives

To understand the following topics:

- Testing and Debugging in PL/SQL
- DBMS_OUTPUT
- UTL_file
- Handling LOB (Large Objects)



DBMS_OUTPUT package

PL/SQL has no input/output capability

However, built-in package DBMS_OUTPUT is provided to generate reports

The procedure PUT_LINE is also provided that places the contents in the buffer

```
PUT_LINE (VARCHAR2 OR NUMBER OR DATE)
```

Displaying Output

Syntax:

```
grant execute on DBMS_LOCK to scott;

SQL>SET SERVEROUTPUT ON
DECLARE
 V_Variable VARCHAR2(25) :=' Used for‘ || 'Debugging ';
BEGIN
DBMS_OUTPUT.PUT_LINE(V_Variable);
END;

BEGIN
dbms_output.put('Hello All,');
dbms_output.put('Good Evening');
dbms_output.put('Today is the 4th day of PLSQL');
dbms_output.new_line();
DBMS_LOCK.sleep(seconds => 10);
END;
/


 BEGIN
 dbms_output.put_line('Hello All,');
  dbms_output.put_line('Good Evening');
  dbms_output.put_line('Today is the 4th day of PLSQL');
  DBMS_LOCK.sleep(seconds => 10);
  END;
   /
Hello All,
Good Evening
Today is the 4th day of PLSQL

PL/SQL procedure successfully completed.
```

DBMS_OUTPUT - Example

In this example, the following anonymous PL/SQL block uses DBMS_OUTPUT to display the name and salary of each staff member in department 10:

```
DECLARE
      CURSOR emp_cur IS SELECT staff_name, staff_sal
       FROM staff_master  WHERE dept_code = 10
            ORDER BY staff_sal DESC;
BEGIN FOR emp_rec IN emp_cur
      LOOP
           DBMS_OUTPUT.PUT_LINE ('Employee ' ||
           emp_rec.staff_name || ' earns ' ||
            TO_CHAR (emp_rec.staff_sal) || 'rupees');
      END LOOP;
      END;
```

UTL_FILE Package

- UTL_FILE package is used for both writing and reading files
- UTL_FILE Process Flow
- **Write to a file:** In order to "write to a file", you will (in most cases) perform the following steps:
    - Declare a file handle. This handle serves as a pointer to the file for subsequent calls to programs in the UTL_FILE package to manipulate the contents of this file
    - Open the file with a call to FOPEN, which returns a file handle to the file. You can open a file to read, replace, or append text
    - Write data to the file by using the PUT, PUTF, or PUT_LINE procedures
    - Close the file with a call to FCLOSE. This releases resources associated with the file

UTL_FILE Process Flow

- **Read from a file:** In order to "read data from a file", you will (in most cases) perform the following steps:
  - Declare a file handle
  - Declare a VARCHAR2 string buffer that will receive the line of data from the file. You can also read directly from a file into a numeric or date buffer. In this case, the data in the file will be converted implicitly, and so it must be compatible with the datatype of the buffer
  - Open the file using FOPEN in read mode
  - Use the GET_LINE procedure to read data from the file and into the buffer. To read all the lines from a file, you would execute GET_LINE in a loop
  - Close the file with a call to FCLOSE

UTL_FILE Example

Create directory sampledata as ‘c:\sampledata’

Above command has to be given by DBA. Command will create a directory object in the database pointing to folder sampledata on c drive of the oracle server machine.

```
declare
f utl_file.file_type;
s varchar2(200);
begin
f := utl_file.fopen('SAMPLEDATA','sample1.txt','R');
utl_file.get_line(f,s);
utl_file.fclose(f);
dbms_output.put_line(s);
end;
```

DBMS_LOB

Handling LOBs (Large Objects)

- Large Objects (LOBs) are a set of datatypes that are designed to hold large amounts of data.
- LOBs are designed to support Unstructured kind of data.
- In short:
  - LOBs are used to store Large Objects (LOBs).
  - LOBs support random access to data and has maximum size of 4 GB
  - For example: Hospital database

Types of LOBs

```
SQL> Create table Leave
  2  (Empno number(4),
  3   S_date date,
  4   E_date date,
  5   snap   blob,
  6   msg    clob);
Table created
```

BLOB- Example

- Setting the LOB to NULL or empty:
  - to set the LOB value to null use the following query:

```
SQL> Insert into Leave values
(7900, '17-APR-98', '20-APR-98', NULL,'The LC and Amendments entry
Forms have been completed. All the validations have been incorporated
and passed for testing.');
1 row created.
```

BLOB - Example

- Setting the LOB to non-NULL:
  - Before writing data to an internal LOB, column must be made NON-NULL, since you cannot call the OCI or the PL/SQL DBMS_LOB functions on a NULL LOB.

```
SQL>  Insert into leave values
  2   (7439,'12-APR-98', '17-APR-98', empty_blob(),
  3   'The assignments regarding Oracle 8 have
  4   been completed. I''ll be back on 17th');
1 row created.
```

Accessing External LOBs

The BFILENAME () function is used to associate a BFILE column with an external file.

- To create a DIRECTORY object:
  - The first parameter to the BFILENAME () function is the directory alias, and the second parameter is the filename.

```
SQL> alter table Leave
add(b_file bfile);
Table altered.
```

BFILE - Example

To create a procedure that displays the first 30 characters of the file, refer the following example:

```
CREATE OR REPLACE PROCEDURE proc_bfile(Eno in number) IS
LOC BFILE;
        V_FILEEXISTS INTEGER;
        v_FILEISOPEN INTEGER;
        NUM NUMBER;
        OFFSET NUMBER;
        LEN NUMBER;
        DIR_ALIAS VARCHAR2(5);
        NAME VARCHAR2(15);
        CONTENTS LONG;
contd.
```

BFILE - Example

```
BEGIN
        SELECT B_FILE INTO LOC FROM LEAVE WHERE EMPNO=Eno;
-- Check to see if file exists
 V_FILEEXISTS := DBMS_LOB.FILEEXISTS(LOC);
 IF v_FILEEXISTS = 1 THEN
        DBMS_OUTPUT.PUT_LINE('The file exists');
 ELSE
 GoTo E;
 END IF;
-- Check if file open
 v_FILEISOPEN := DBMS_LOB.FILEISOPEN( LOC );
contd.
```

BFILE - Example

```
--Determine actions if file is opened or not
 IF v_FILEISOPEN = 1 THEN
       DBMS_OUTPUT.PUT_LINE('The file is open');
 ELSE
       DBMS_OUTPUT.PUT_LINE('Opening the file');
       DBMS_LOB.FILEOPEN(LOC);
       LEN := DBMS_LOB.GETLENGTH( LOC );
       DBMS_OUTPUT.PUT_LINE('Length of the file : ' ||
       TO_CHAR(LEN));
       NUM := 40; OFFSET := 1;
       DBMS_LOB.READ(LOC, NUM, OFFSET, CONTENTS);
contd.
```

BFILE - Example

```
      DBMS_OUTPUT.PUT_LINE('Contents of the file : '||
 CONTENTS);
END IF;
DBMS_LOB.FILEGETNAME(LOC, DIR_ALIAS, NAME);
DBMS_OUTPUT.PUT_LINE ('Opening ' || dir_alias ||
name);
DBMS_LOB.FILECLOSE(LOC); -- Close the BFILE
<<E>>
DBMS_OUTPUT.PUT_LINE('The file cannot be found');
END;  /
```

BFILE - Example

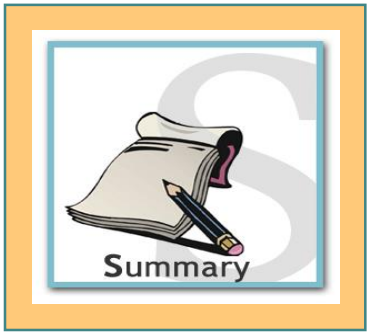
- Test the procedure by executing it at SQL prompt as follows:
  - MSG: PL/SQL procedure successfully completed.

```
SQL>  execute proc_bfile(7900);
The file exists
Opening the file
Length of the file : 30
Contents of the file: BOSTONS MANAGEMENT CONSULTANTS
Opening L_DIR TEST.TXT
```

Summary

In this lesson, you have learnt about:

- Testing and Debugging in PL/SQL
- DBMS_OUTPUT
- Enabling and Disabling output
- Writing to the DBMS_OUTPUT Buffer
- UTL_file
- Handling LOB (Large Objects)



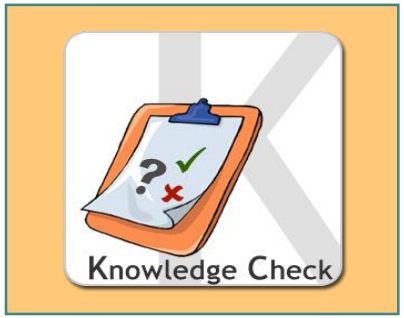
Review Questions

Question 1: The value held in a LOB column or variable is not the actual binary data, but a "locator" or pointer to the physical location of the large object.

- True / False

Question 2: CLOB stores a column that supports multi-byte characters from National Character set defined by Oracle.

- True / False



Review Questions

Question 3: If the pointer to the file is already located at the last line of the file, UTL_FILE.GET_LINE does not return data.

- True / False

Question 4: The file can be open in one of the following three modes: ____, ____, and ____.

Question 5: The package ____ lets you read and write files accessible from the server on which your database is running.