NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCES

CL 203-Database Systems Lab

Lab Session 10

Loops

1. Simple Loops	Run until you explicitly end the loop
2. **WHILE Loops**	Run until a specified condition occurs
3. FOR Loops	Run a predetermined number of times

1) Simple Loops

A simple loop runs until you explicitly end the loop.

```
LOOP
        Statements
END LOOP;
```

2) WHILE Loops

A While loop runs until a specified condition occurs.

```
        WHILE condition LOOP
                Statements
        END LOOP;
```

Ex:

```
Counter   :=0;
WHILE counter <6   LOOP
        Counter   := counter +1;
END LOOP;
```

3) FOR Loops

A FOR loop runs a predetermined number of times; you determine the number of times the loop runs by specifying the lower and upper bounds for a loop variable . The loop variable is then incremented (or decremented) each time around the loop. The syntax for a FOR loop is as follows:

FOR   Loop_variable IN***lower_bound . . upper_bound***   LOOP
        Statements

```
END LOOP;
```

Ex:

```
FOR count2  IN  1 . . 5 LOOP
DBMS_OUTPUT.PUT_LINE (count2);
END LOOP;
```

Cursors

You use cursor when you have a SELECT statement that returns more than one row from the database. A cursor is basically a set of rows that you can access one at a time. You may follow five steps when using a cursor:

1. **Declare variables to store column values from the SELECT statement.**
2. **Declare the cursor, specifying your SELECT statement.**
3. **Open the Cursor.**
4. **Fetch the rows from the cursor.**
5. **Close the cursor.**

STEP 1 : Declare variables to store column values

These variables must be compatible with the column types.

```
DECLARE
v_Empno emp.empno%TYPE;
v_Ename emp.ename%TYPE;
v_Job emp.job%TYPE;
v_Deptno emp.deptno%TYPE;
v_Sal    emp.sal%TYPE;
```

STEP 2: **Declare the cursor**

```
CURSOR   cursor_name  IS

SELECT_STATEMENT;

---
CURSOR    cv_emp_cursor      IS
SELECT empno,ename,sal
FROM products
Order by empno;
```

STEP 3: Open the Cursor

This step runs the SELECT statement. It must be placed in the executable section of the block (BEGIN).

```
OPEN cv_emp_cursor;
```

STEP 4: Fetch the rows from the cursor

To read each row from the cursor, you can use the FETCH statement. The FETCH statement reads the column values into the variables that you specify.

```
FETCH  cursor_name
INTO      variable[,  variable. . . . ];
```

```
FETCH cv_emp_cursor
INTO     v_Empno,v_Ename   ,  v_Job     ,      v_Deptno    ,v_Sal;
```

A cursor may have many rows; therefore, a loop is required to read each row in turn.

```
LOOP

FETCH CV_EMP_CURSOR
INTO  V_EMPNO    ,V_ENAME   ,  V_JOB     ,      V_DEPTNO    ,V_SAL;
--exit the loop when there are no more rows, as indicated by
--the Boolean variable cv_emp_cursor%NOTFOUND (=true when
--there are no more rows)
EXIT WHEN cv_emp_cursor%NOTFOUND;
--use DBMS_OUTPUT.PUT_LINE () to display the variable
DBMS_OUTPUT.PUT_LINE(
‘v_Empno= ‘ || v_Empno   ,’v_Ename=’ || v_ename   ,’  v_Job= ‘ || v_job     ,    ‘ v_Deptno= ‘
||  v_Deptno   ,’v_Sal = ‘|| v_sal;
```

STEP 5: Close the Cursor

Closing your cursors frees up system resources.

```
CLOSE   cv_emp_cursor;
```

Assemble the above code together so that you can run it on SQL* PLUS:

Cursors and for loops

You can use the power of FOR loop to access the rows in a cursor. When you use a FOR loop, you don’t have to explicitly open and close the cursor------ the FOR loop does this automatically.

```
DECLARE
   CURSOR cv_emp_cursor2 IS
      SELECT empno, ename,job
        FROM emp WHERE deptno = 30;
BEGIN
   FOR v_emp IN cv_emp_cursor2 LOOP
      DBMS_OUTPUT.PUT_LINE(
               ‘EMPNO =’ || v_emp.empno || ‘,ename = ‘|| v_emp.ename || ‘,job = ‘|| v_emp.job );
END LOOP;
END;
/
```

Exceptions

Exceptions are used to handle errors that occur in your PL/SQL code. Following are some exceptions occur in code

Exception	**Raised when ...**
ACCESS_INTO_NULL	Your program attempts to assign values to the attributes of an uninitialized (atomically null) object.
CASE_NOT_FOUND	None of the choices in the WHEN clauses of a CASE statement is selected, and there is no ELSE clause.
COLLECTION_IS_NULL	Your program attempts to apply collection methods other than EXISTS to an uninitialized (atomically null) nested table or varray, or the program attempts to assign values to the elements of an uninitialized nested table or varray.
CURSOR_ALREADY_OPEN	Your program attempts to open an already open cursor. A cursor must be closed before it can be reopened. A cursor FOR loop automatically opens the cursor to which it refers. So, your program cannot open that cursor inside the loop.
DUP_VAL_ON_INDEX	Your program attempts to store duplicate values in a database column that is constrained by a unique index.
INVALID_CURSOR	Your program attempts an illegal cursor operation such as closing an unopened cursor.
INVALID_NUMBER	In a SQL statement, the conversion of a character string into a number fails because the string does not represent a valid number. (In procedural statements, VALUE_ERROR is raised.) This exception is also raised when the LIMIT-clause expression in a bulk FETCH statement does not evaluate to a positive number.
LOGIN_DENIED	Your program attempts to log on to Oracle with an invalid username and/or password.
NO_DATA_FOUND	A SELECT INTO statement returns no rows, or your program

Exception	Raised when ...
	references a deleted element in a nested table or an uninitialized element in an index-by table. SQL aggregate functions such as AVG and SUM always return a value or a null. So, a SELECT INTO statement that calls an aggregate function never raises NO_DATA_FOUND. The FETCH statement is expected to return no rows eventually, so when that happens, no exception is raised.
NOT_LOGGED_ON	Your program issues a database call without being connected to Oracle.
PROGRAM_ERROR	PL/SQL has an internal problem.
ROWTYPE_MISMATCH	The host cursor variable and PL/SQL cursor variable involved in an assignment have incompatible return types. For example, when an open host cursor variable is passed to a stored subprogram, the return types of the actual and formal parameters must be compatible.
SELF_IS_NULL	Your program attempts to call a MEMBER method on a null instance. That is, the built-in parameter SELF (which is always the first parameter passed to a MEMBER method) is null.
STORAGE_ERROR	PL/SQL runs out of memory or memory has been corrupted.
SUBSCRIPT_BEYOND_COUNT	Your program references a nested table or varray element using an index number larger than the number of elements in the collection.
SUBSCRIPT_OUTSIDE_LIMIT	Your program references a nested table or varray element using an index number (-1 for example) that is outside the legal range.
SYS_INVALID_ROWID	The conversion of a character string into a universal rowid fails because the character string does not represent a valid rowid.
TIMEOUT_ON_RESOURCE	A time-out occurs while Oracle is waiting for a resource.
TOO_MANY_ROWS	A SELECT INTO statement returns more than one row.
VALUE_ERROR	An arithmetic, conversion, truncation, or size-constraint error occurs. For example, when your program selects a column value into a character variable, if the value is longer than the declared length of the variable, PL/SQL aborts the assignment and raises VALUE_ERROR. In procedural statements, VALUE_ERROR is raised if the conversion of a character string into a number fails. (In SQL statements, INVALID_NUMBER is raised.)
ZERO_DIVIDE	Your program attempts to divide a number by zero.

ACTIVITY

1) Write a PL/SQL block that finds that the number is Armstrong or not?
   What is Armstrong Number?
   If the sum of cube of digits of a number forms the same number then it is known as Armstrong Number.
   Example: 371 = (3*3*3) + (7*7*7) + (1*1*1) = 27 + 343 + 1 = 371

2) Write a PL/SQL that finds the sum of digits of number.

3) Write a PL/SQL block that generates a Fibonacci series.

   The first two numbers in the Fibonacci series are 0 and 1, and each subsequent number is the sum of the previous two.
   0,1,1,2,3,5,8,13,21,34,

4) Write a PL/SQL block that finds the given string is palindrome or not??

5) Write a cursor to display list of employees and total salary department wise.

6) Write a cursor to display list of employees who are working as manager.

7) Write a cursor to display an employee with job and department.

8) Write a cursor to display list of employees.

BEST OF LUCK