Qatar University
College of Engineering
Computer Science & Engineering Dept.

Fundamentals of Database Systems

Lab 8

Objectives:

At the end of this lab, you should be able to:
Write simple Procedural Language/SQL (PL/SQL) program.

Procedural Language/SQL (PL/SQL) is an extension of Structured Query Language (SQL) that is used in Oracle. Unlike SQL, PL/SQL allows the programmer to write code in a procedural format.

PL/SQL is a block-structured language, meaning that programs can be divided into logical blocks. A PL/SQL block consists of up to three sections:

- **Declarative (optional):** contains all variables, constants, coursers that are used in the executable section.
- **Executable (required):** contains SQL statements to manipulate data in the database and PL/SQL statements to manipulate data in the block.
- **Exception handling (optional):** specifies the action to perform when errors and abnormal conditions arise in the executable section.

Declarative

```
DECLARE
 V       VARCHAR2(5);
```

Executable

```
BEGIN
 SELECT     column_name
   INTO     v
   FROM     table_name;
```

Exception handling

```
EXCEPTION
 WHEN exception_name THEN
 ...
END;
```

Declaring PL/SQL Variables

```
Declare
        v_hiredate          DATE;
        v_deptno            NUMBER(2) NOT NULL := 10;
        v_location          VARCHAR2(13) := 'Atlanta';
       c_comm               CONSTANT NUMBER := 1400;
```

Initialize the variable to an expression with the assignment operator (:=) or, equivalently, with the DEFAULT reserved word. If you do not assign an initial value, the new variable contains NULL by default until you assign it later.

The %TYPE Attribute

You can use the %TYPE attribute to declare a variable according to another previously declared variable or database column

```
...
  v_ename                   emp.ename%TYPE;
  v_balance                 NUMBER(7,2);
  v_min_balance             v_balance%TYPE := 10;
...
```

The %ROWTYPE Attribute

- Declare a variable according to a collection of columns in a database table or view.
- Prefix %ROWTYPE with the database table.
- Fields in the record take their names and datatypes from the columns of the table or view.

For example:

dept_record dept%ROWTYPE;

To access a field of the variable of dept_record, you use dot.

dept_record.depno=50;

Boolean Variables

- Only the values TRUE, FALSE, and NULL can be assigned to a Boolean variable.
- The variables are connected by the logical operators **AND**, **OR**, and **NOT.**

V_sal Boolean :=true;

Writing Executable Statements

Because PL/SQL is an extension of SQL, the general syntax rules that apply to SQL also apply to the PL/SQL language.

1. Commenting Code

Comment the PL/SQL code with two dashes (--) if the comment is on a single line, or enclose the comment between the symbols /* and */ if the comment spans several lines

2. Retrieving Data Using PL/SQL

- Use the SELECT statement to retrieve data from the database with into clause. The INTO clause is mandatory and occurs between the SELECT and FROM clauses.
- You must give one variable for each item selected, and their order must correspond to the items selected.
- Queries Must Return One and Only One Row.

```
DECLARE
  v_deptno    NUMBER(2);
  v_loc       VARCHAR2(15);
BEGIN
  SELECT      deptno, loc
  INTO        v_deptno, v_loc
  FROM        dept
  WHERE       dname = 'SALES';
...
END;
```

To display information from a PL/SQL block you can use *DBMS_OUTPUT.PUT_LINE*. DBMS_OUTPUT is an Oracle-supplied package, and PUT_LINE is a procedure within that package. Within a PL/SQL block, reference DBMS_OUTPUT.PUT_LINE and, in parentheses, the information you want to print to the screen. The package must first be enabled in your session. To do this, execute the command *SET SERVEROUTPUT ON.*

Example:

```
SET SERVEROUTPUT ON
DECLARE
  v  NUMBER(6,2) ;
BEGIN
SELECT AVG(SAL) INTO v FROM EMP;

  DBMS_OUTPUT.PUT_LINE ('The average salary is ' || TO_CHAR(v));
END;
```

Manipulating Data Using PL/SQL

You can issue the DML commands INSERT, UPDATE, and DELETE without restriction in PL/SQL. Including COMMIT or ROLLBACK statements

```
DECLARE
  v_sal_increase   emp.sal%TYPE := 2000;
BEGIN
  UPDATE      emp
  SET         sal = sal + v_sal_increase
  WHERE       job = 'ANALYST';
END;
```

Note: PL/SQL variable assignments always use := and SQL column assignments always use =.

Writing Control Structures

1. IF Statements

Syntax :

```
IF condition THEN
  statements;
[ELSIF condition THEN
  statements;]
[ELSE
  statements;]
END IF;
```

For example:

write a program that displays number of employees whose salaries is less than 5000 or display a message “ No employee found”

```
Declare
            x  number(3):=0;
Begin
        Select count(*) into x  from emp where sal<5000;
        If x=0 then
                Dbms_output.put_line(‘No employee found’);
        Else
                Dbms_output.put_line(‘# of employees ‘ || x);
        End if;
End;
```

Exception

```
Declare
            x  number(3):=0;
Begin
        Select deptno into x from dept where dname=’IT’;
Exception
When no_data_found then
        Dbms_output.put_line(‘ No such department’);
End;
```

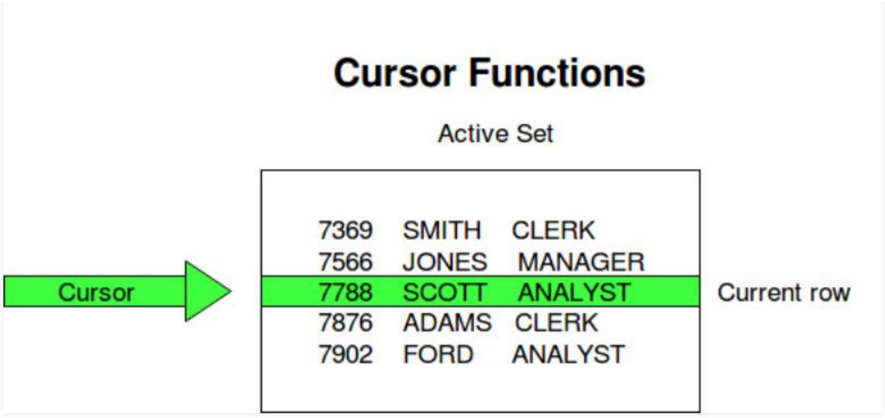
2. <u>FOR LOOP</u>

<u>Example # 1:</u>

```
SET SERVEROUTPUT ON;
BEGIN
FOR counter_v IN reverse 1 .. 10
LOOP
        dbms_output.put_line(counter_v);
END LOOP;
END;
```

Use of Cursor:

The major function of a cursor is to retrieve data, one row at a time. The Data that is stored in the Cursor is called the Active Data Set. Oracle DBMS has another predefined area in the main memory Set, within which the cursors are opened.



Cursor Functions

Active Set

7369	SMITH	CLERK
7566	JONES	MANAGER
7788	SCOTT	ANALYST
7876	ADAMS	CLERK
7902	FORD	ANALYST

There are four steps in using an Explicit Cursor.

- DECLARE the cursor in the Declaration section.
- OPEN the cursor in the Execution Section.
- FETCH the data from the cursor into PL/SQL variables or records in the Execution Section.
- CLOSE the cursor in the Execution Section before you end the PL/SQL Block.

Example # 2:

```
SET SERVEROUTPUT ON;
DECLARE
  CURSOR CUR IS
  SELECT EMPNO,ENAME
  FROM EMP
  WHERE SAL > 2000;
BEGIN
FOR I IN CUR
LOOP
  DBMS_OUTPUT.PUT_LINE(I.EMPNO||'  '||I.ENAME);
END LOOP;
END;
```

EXERCISE:

- Write a PL-SQL Script to display the Job and department name of the employee JAMES on the screen.
- Modify the above program to display the Names, Jobs, and Department Names for all employees.
- Write a PL-SQL Script to give a salary increase of 5% to JAMES. Print on the Screen the Total Salary for JAMES after Salary increase.
- Write a PL-SQL Script that lists all employees' names and salary from the employee table using a cursor.
  Print in the following format - Example:
  ALLEN HAS A SALARY OF 1600 USD
- Write a PL-SQL Script that will produce the following output:
  DEPTNO = 20
  DNAME = RESEARCH
  LOC = DALLAS
  DEPARTMENT AVERAGE SALARY = 2175