DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING
MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO
Database Management Systems (4th Semester) 18CS
Lab Experiment 11

Roll No: **Date of Conduct:**

Submission Date: **Grade Obtained:**

Problem Recognition (0.3)	Completeness & accuracy (0.4)	Timeliness (0.3)	Score (1.0)

Objective: **To understand and practice PL/SQL block structure, Control Structures and Data Types.**

Tools: MySql, Oracle.

Introduction:

PL/SQL Block Structure

In PL/SQL, as in most other procedural languages, the smallest meaningful grouping of code is known as a block. A block is a unit of code that provides execution and scoping boundaries for variable declarations and exception handling. PL/SQL allows you to create anonymous blocks (blocks of code that have no name) and named blocks, which may be packages, procedures, functions, triggers, or object types..

A PL/SQL block has up to four different sections, only one of which is mandatory:

Header

Used only for named blocks. The header determines the way the named block or program must be called. Optional.

Declaration section

Identifies variables, cursors, and subblocks that are referenced in the execution and exception sections. Optional.

Execution section

Statements the PL/SQL runtime engine will execute at runtime. Mandatory.

Exception section

Handles exceptions to normal processing (warnings and error conditions). Optional

Basic Syntax structure

Declare

Declaration of variable, constants

Begin

Execute statements in pl/sql

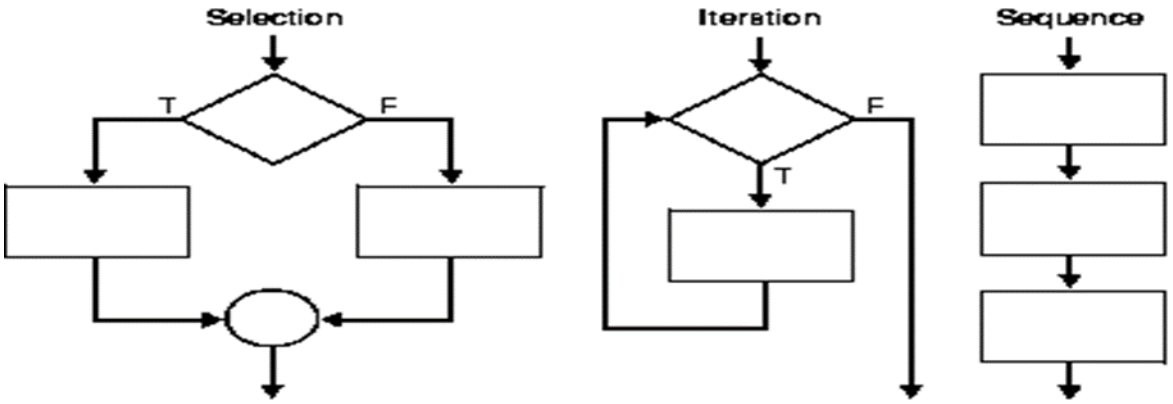
Exception

Exception handlers in pl/sql

END;

PL/SQL Control Structures

The selection structure tests a condition, then executes one sequence of statements instead of another, depending on whether the condition is true or false. A condition is any variable or expression that returns a Boolean value (TRUE or FALSE). The iteration structure executes a sequence of statements repeatedly as long as a condition holds true. The sequence structure simply executes a sequence of statements in the order in which they occur.

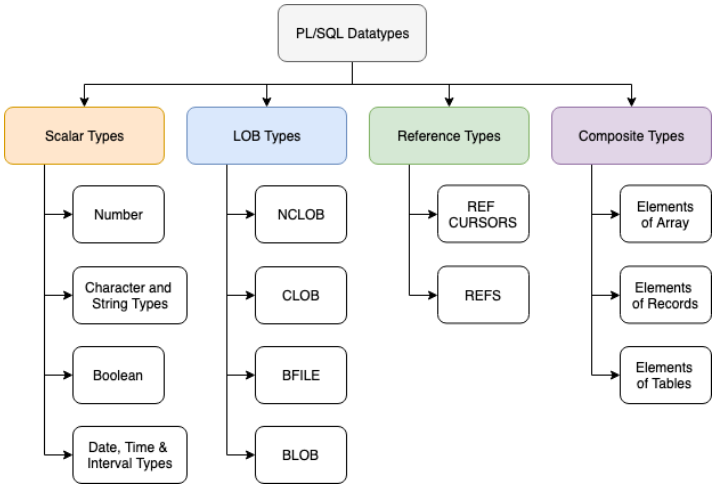


PL/SQL Data Types.

PL/SQL datatypes are not just limited to writing SQL queries but they are used in the PL/SQL block as well, just like any other programming language.

Provising a datatype specifies how any data will be stored and processed by Oracle when any PL/SQL code block is executed.

Datatype defines the type of data being used, whether it is a number or a word (string) or a single character etc. Following datatypes can be used in PL/SQL depending upon the type of data required:



So we have 4 broader categories of datatypes and they are:

1. Scalar Types: These are basic datatypes which generally holds a single value like a number or a string of characters. Scalar types have 4 different categories which are listed in the diagram above, namely Number Types, Character and String, Boolean Types and Date and Time etc.

2. **LOB Types**: This datatype deals with large objects and is used to specify location of these large objects like text files, images etc which are generally not stored outside the database.

3. **Reference Types:** This datatype is used to hold pointer values which generally stores address of other program items.

4. **Composite Types**: Last but not the least, as the name suggests this type of data is a composition of individual data which can be manipulated/processed separatel as well.

LAB TASK

1. **Write a PL/SQL block to calculate the annual salary of an employee whose ID is 7722.**

Task:

```
SET SERVEROUTPUT ON;
DECLARE
  incentive   NUMBER(8,2);
BEGIN
  SELECT SAL * 0.12 INTO incentive
  FROM EMP
  WHERE EMPNO = 7722;
DBMS_OUTPUT.PUT_LINE('Incentive  = ' || TO_CHAR(incentive));
END;
```

Script Output | Query Result | Task completed in 0.07 seconds

```
Incentive  = 360


PL/SQL procedure successfully completed.
```

2. **Write a PL/SQL block to show the operator precedence and parentheses in 5 or more complex expressions.**

Task:

```
DECLARE
  salary      NUMBER := 40000;
  commission  NUMBER := 0.15;
BEGIN
  -- Division has higher precedence than addition:

  DBMS_OUTPUT.PUT_LINE('8 + 20 / 4 = ' || (8 + 20 / 4));
  DBMS_OUTPUT.PUT_LINE('20 / 4 + 8 = ' || (20 / 4 + 8));

 -- Parentheses override default operator precedence:

  DBMS_OUTPUT.PUT_LINE('7 + 9 / 3 = ' || (7 + 9 / 3));
  DBMS_OUTPUT.PUT_LINE('(7 + 9) / 3 = ' || ((7 + 9) / 3));

  -- Most deeply nested operation is evaluated first:

  DBMS_OUTPUT.PUT_LINE('30 + (30 / 6 + (15 - 8)) = '
                       || (30 + (30 / 6 + (15 - 8))));

  -- Parentheses, even when unnecessary, improve readability:

  DBMS_OUTPUT.PUT_LINE('(salary * 0.08) + (commission * 0.12) = '
    || ((salary * 0.08) + (commission * 0.12)));

  DBMS_OUTPUT.PUT_LINE('salary * 0.08 + commission * 0.12 = '
    || (salary * 0.08 + commission * 0.12));
END;
```

Script Output | Query Result | Task completed in 0.051 seconds

```
8 + 20 / 4 = 13
20 / 4 + 8 = 13
7 + 9 / 3 = 10
(7 + 9) / 3 = 5.333333333333333333333333333333333333333
30 + (30 / 6 + (15 - 8)) = 42
(salary * 0.08) + (commission * 0.12) = 3200.018
salary * 0.08 + commission * 0.12 = 3200.018
```

3. **Write a PL/SQL program to arrange the number of two variable in such a way that the small number will store in num_small variable and large number will store in num_large variable.**

Task:

```
DECLARE
num_small NUMBER := 8;
num_large NUMBER := 5;
num_temp NUMBER;
BEGIN

IF num_small > num_large THEN
num_temp := num_small;
num_small := num_large;
num_large := num_temp;
END IF;

DBMS_OUTPUT.PUT_LINE ('num_small = '||num_small);
DBMS_OUTPUT.PUT_LINE ('num_large = '||num_large);
END;
```

Script Output | Query Result | Task completed in 0.045 seconds

```
num_small = 5
num_large = 8


PL/SQL procedure successfully completed.
```

4. **Write a PL/SQL program to count number of employees in department 30 and check whether this department have any vacancies or not. There are total 45 posts in this department.**

Task:

```
DECLARE
    CURSOR z_emp_info IS
      SELECT deptno,ename,sal
      FROM   emp where deptno=30;
    r_emp_info z_emp_info%ROWTYPE;
BEGIN
    OPEN z_emp_info;
    LOOP
        FETCH z_emp_info INTO r_emp_info;
        EXIT WHEN z_emp_info%NOTFOUND;
        dbms_output.Put_line('Name: '||r_emp_info.ename
                             ||'  SALARY: ' ||r_emp_info.sal||
                             '  DEPTNO: ' ||r_emp_info.deptno);
    END LOOP;
    dbms_output.Put_line('Total number of rows : '||z_emp_info%rowcount);
    CLOSE z_emp_info;
```

Script Output | Explain Plan | Task completed in 0.077 seconds

```
Name: BLAKE  SALARY: 3850  DEPTNO: 30
Name: ALLEN  SALARY: 2600  DEPTNO: 30
Name: WARD  SALARY: 2250  DEPTNO: 30
Name: MARTIN  SALARY: 2250  DEPTNO: 30
Name: TURNER  SALARY: 2500  DEPTNO: 30
Name: JAMES  SALARY: 1950  DEPTNO: 30
Total number of rows : 6


PL/SQL procedure successfully completed.
```

5. **Write a program in PL/SQL to check whether a number is prime or not using for loop.**

Task:

```
DECLARE
     n NUMBER := 17;
     i NUMBER:= 2 ;
     flag NUMBER:= 1;
BEGIN

 FOR i IN 2..n/2 LOOP
   IF  MOD(n,i)=0 THEN
          flag:=0; EXIT;
        END IF;
    END LOOP;

IF flag=1 THEN
        dbms_output.put_line(n||' is Prime Number');
    ELSE
        dbms_output.put_line(n||'is not Prime  Number');
    END IF;
END;
```

Script Output × Query Result ×

Task completed in 0.038 seconds

```
17 is Prime Number


PL/SQL procedure successfully completed.
```