Chapter 15

INTRODUCTION TO PL/SQL

- What is PL/SQL?
- PL/SQL engine
- Features of PL/SQL
- Advantages of PL/SQL
- PL/SQL Block
- Writing first PL/SQL block
- Nested blocks
- Scope and visibility of variables
- Labeling the block
- Assignment Operator
- Displaying output from PL/SQL

What is PL/SQL?

PL/SQL is Oracle's procedural language extension to SQL. PL/SQL allows you to mix SQL statements with procedural statements like IF statement, Looping structures etc. PL/SQL is the superset of SQL. It uses SQL for data retrieval and manipulation and uses its own statements for data processing.

PL/SQL program units are generally categorized as follows:

- Anonymous blocks
- Stored procedures

Anonymous block

This is a PL/SQL block that appears within your application. In many applications PL/SQL blocks can appear where SQL statements can appear. Such blocks are called as *Anonymous blocks*.

Stored Procedure

This is a PL/SQL block that is stored in the database with a name. Application programs can execute these procedures using the name. Oracle also allows you to create functions, which are same as procedures but return a value, and packages, which are a collection of procedures and functions.

PL/SQL Engine

Every PL/SQL block is first executed by PL/SQL engine. This is the engine that compiles and executes PL/SQL blocks. PL/SQL engine is available in Oracle Server and certain Oracle tools such as Oracle Forms and Oracle Reports.

PL/SQL engine executes all procedural statements of a PL/SQL of the block, but sends SQL command to ***SQL statements executor*** in the Oracle RDBMS. That means PL/SQL separates SQL commands from PL/SQL commands and executes PL/SQL commands using ***Procedural statement executor***, which is a part of PL/SQL engine. See figure 1.

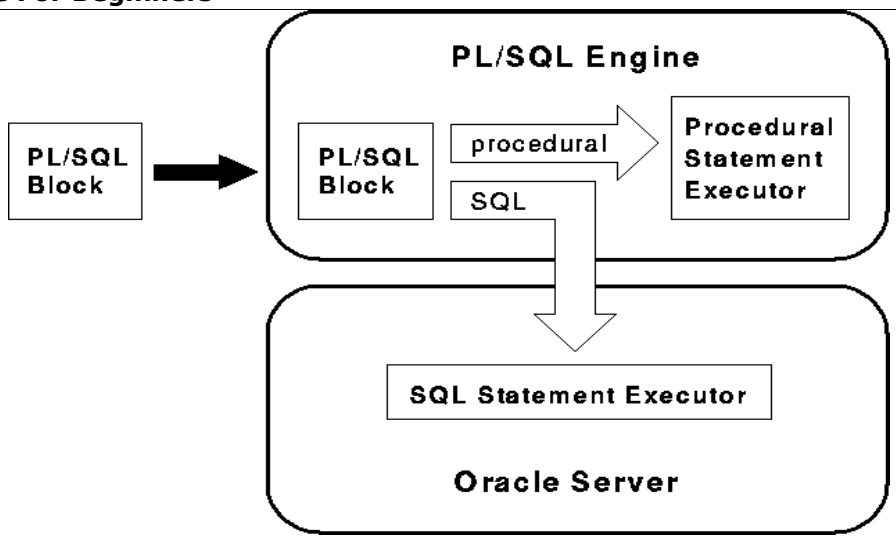


Figure 1: PL/SQL Engine.

Features of PL/SQL

The following are important features of PL/SQL.

Block structure

PL/SQL is a block-structured language. Each program written in PL/SQL is written as a block. Blocks can also be nested. Each block is meant for a particular task.

Variables and constants

PL/SQL allows you to declare variables and constants. Variables are used to store values temporarily. Variables and constants can be used in SQL and PL/SQL procedural statements just like an expression.

Control structures

PL/SQL allows control structures like IF statement, FOR loop, WHILE loop to be used in the block. Control structures are most important extension to SQL in PL/SQL. Control structures allow any data process possible in PL/SQL.

Exception handling

PL/SQL allows errors, called as exceptions, to be detected and handled. Whenever there is a predefined error PL/SQL raises an exception automatically. These exceptions can be handled to recover from errors.

Modularity

PL/SQL allows process to be divided into different modules. Subprograms called as procedures and functions can be defined and invoked using the name. These subprograms can also take parameters.

Cursors

A cursor is a private SQL area used to execute SQL statements and store processing information. PL/SQL implicitly uses cursors for all DML commands and SELECT command that returns only one row. And it also allows you to define explicit cursor to deal with multiple row queries.

Built-in functions

Most of the SQL functions that we have seen so far in SQL are available in PL/SQL. These functions can be used to manipulate variables of PL/SQL.

Advantages Of PL/SQL

The following are a few important advantages of PL/SQL. Moreover most of the features listed above are also advantages of PL/SQL.

Support for SQL

PL/SQL allows you to use SQL commands, function and operators. PL/SQL supports data types of SQL.

PL/SQL also allows SQL statements to be constructed and executed on the fly. The process of creating SQL statements on the fly is called as **Dynamic SQL**. This is different from writing SQL commands at the time of writing the program, which is called as **Static SQL**.

Starting from Oracle8i, PL/SQL support native dynamic SQL, which makes programming Dynamic SQL easier than its predecessor, where we used DBMS_SQL package.

We will see more about Dynamic SQL in later chapter.

Better performance

PL/SQL block is sent as one unit to Oracle server. Without PL/SQL each SQL command is to be passed to Oracle server, which will increase network traffic heavily. As a collection of SQL statements is passed as a block to Oracle server, it improves performance.

Portability

Applications written in PL/SQL are portable to any platform on which Oracle runs. Once you write a program in PL/SQL, it can be used in any environment without any change at all.

PL/SQL block

PL/SQL programs are written as blocks. Block allows you to group logically related statements and declarations. PL/SQL block is consisting of the following three parts:

- Declarative part
- Executable part
- Exception-handling part

The following is the syntax of PL/SQL block.

```
[DECLARE

      declaration of  variable
      declaration of cursor
      declaration of  exception  ]

BEGIN
      executable commands

[EXCEPTION

      exception  handlers]

END;
```

Declarative Part

This is the area of the block where variables, cursors etc are declared. All variables used in the block are to be declared in declarative part.

The following is the example of declarative part. First variable is of type NUMBER(5). As we have seen before PL/SQL supports the data types of SQL. The second variable is initialized to 0.

```
declare
     v_rollno  number(5);
     v_count   number(2) := 0;
     v_name    students.name%type;
     done      boolean      := FALSE;
```

Variable V_NAME is declared as of type STUDENTS.NAME column. Attribute %TYPE takes whatever is the data type of NAME column of STUDENTS table and uses the same data type to declare V_NAME variable.

***Note**: If the name of a variable and the name of a column are same then Oracle assumes that the column is being referenced whenever the name is used in SQL commands. That means columns names takes precedence over variable name in SQL statements.*

Declarative block can also be used to declare CURSORS and EXCEPTIONS. But, I feel they are going to be heavy for now so we will defer their discussion until later chapters.

PL/SQL Datatypes

PL/SQL provides a variety of predefined datatypes, which can be divided into four categories:

Scalar Represents a single value.
Composite Is a collection of components
Reference Is a pointer that points to another item.
LOB Holds a lob locator.

The following are the datatypes in various categories

Scalar	NUMBER, CHAR, VARCHAR2, DATE, BOOLEAN
Composite	RECORD, TABLE and VARRAY.
Reference	REF CURSOR, REF Object_type
LOB	BFILE, BLOB, CLOB, and NCLOB.

***Note:** There may be minor differences between PL/SQL datatypes and SQL datatypes though they have the same name. For complete information about datatypes in PL/SQL please see PL/SQL User's guide and Reference.*

Executable part

Is the area where we write SQL and PL/SQL commands that are to be executed. This is the only mandatory part of the entire block.

Exception-handling part

Is the place where we handle exceptions (errors) that are raised in executable part. Exception handlers handle exceptions. We will discuss more about this in later chapter.

Writing first PL/SQL block

The best way to get accustomed to PL/SQL is by writing a few blocks. Let us write our first PL/SQL block. PL/SQL block used here is more to highlight the syntax of a PL/SQL block rather than to show what you can do with PL/SQL block that you cannot do with SQL.

```
declare

    v_rollno  students.rollno%type;

begin

    -- get roll number of the students who joined most recently

    select max(rollno) into  v_rollno
    from students;

    -- insert a new row into payments table

    insert into payments values (v_rollno,sysdate,1000);

    -- commit transaction

    commit;

end;
/
```

Follow the procedure given below to create and run the above block.

1. Type the above program in a text editor such as Notepad and save it in a text file. Assume the file is saved under the name INSPAY.SQL. And be sure to know the directory where the file is saved.

   Note: *Enclose filename in double quotes at the time of giving file name in Notepad, otherwise Notepad will add the extension .TXT to the file.*

2. Get into SQL*PLUS. Start it and logon if you have not already logged on.
3. Use START command to execute the program that is in the file INSPAY.SQL.

```
SQL> start c:\orabook\inspay.sql

PL/SQL procedure successfully completed.
```

If the block is successfully executed then PL/SQL displays the above message. If there are any errors during the execution then you have to correct the program, save the program and rerun it until you succeed.

Comments in PL/SQL

You can give comments in PL/SQL block in two ways.

First way is by preceding the comment with two hyphens (- -).

Example: -- this is single line comment

Second way is by starting the comment with /* and ending it with */.

Example: /* this comment can be of multiple lines */

SELECT… INTO

SQL*Plus displays the data retrieved by SELECT command. Whereas in PL/SQL SELECT command is used only to retrieve the data and storing and using data is to be done explicitly. So Oracle provided INTO clause with SELECT command that is used to specify the variable(s) into which the value(s) retrieved must be copied.

The following example copies the total amount paid by student with roll number 102 to variable V_SUM, which is declared as NUMBER(5) data type.

```
declare
     v_sum number(5);
     v_fee courses.fee%type;
     v_dur courses.duration%type;
begin
     select  sum(amount) into v_sum
     from payments
     where rollno = 102;

     -- take fee and duration of Ora course

     select fee, duration into v_fee, v_duration
     from   courses
     where  ccode = 'ora';

     . . .

end;
```

The number of variables and data types given after INTO clause must match the number of columns in SELECT and their data types. In the second example, SELECT selects two columns (fee, duration) so INTO clause must have two variables (v_fee, f_duration).

Declaring Constants

Constant is a PL/SQL variable whose value doesn't change. Any attempt to change the value of a constant will result in error.

```
variable CONSTANT datatype [precision , scale] := expression;
```

The following declarative statement creates a constant that takes value 500.

```
bonus constant  number(3) :=  500;
```

Nesting Blocks

It is possible to define a block within another block. When blocks are defined one within another they are said to be nested.

The following is an example of nested block. Main block contains a single variable X and nested block contains a single variable Y.

```
declare
    x number(5);
begin

      -- put executable code of  main  block here

      declare     /* beginning  of  nested block */
         y number(3);
      begin
            --  put executable code of  nested block
      exception
            --  exception handing  for  nested block
      end;

     -- code of main block continues.

Exception
```

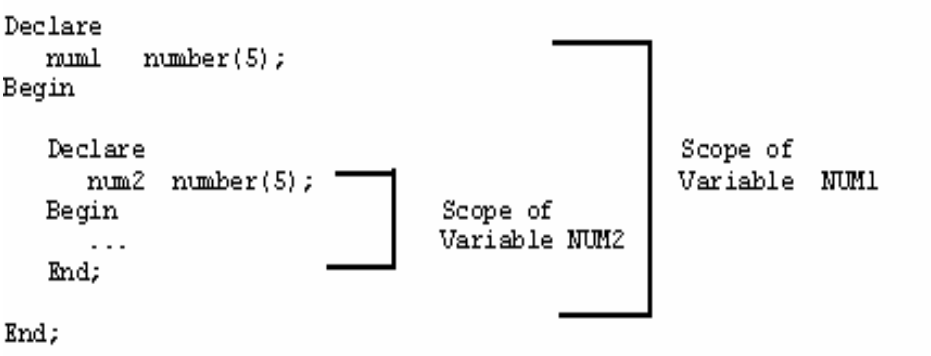
```
      --  exception handling for main block
end;
```

Scope and visibility of variables

Scope of the variable refers to the region of the program in which the variable can be used. A variable is said to be *visible* when it can be referred without any qualifier.

Examine the following examples to understand scope and visibility of variables.

Example 1:

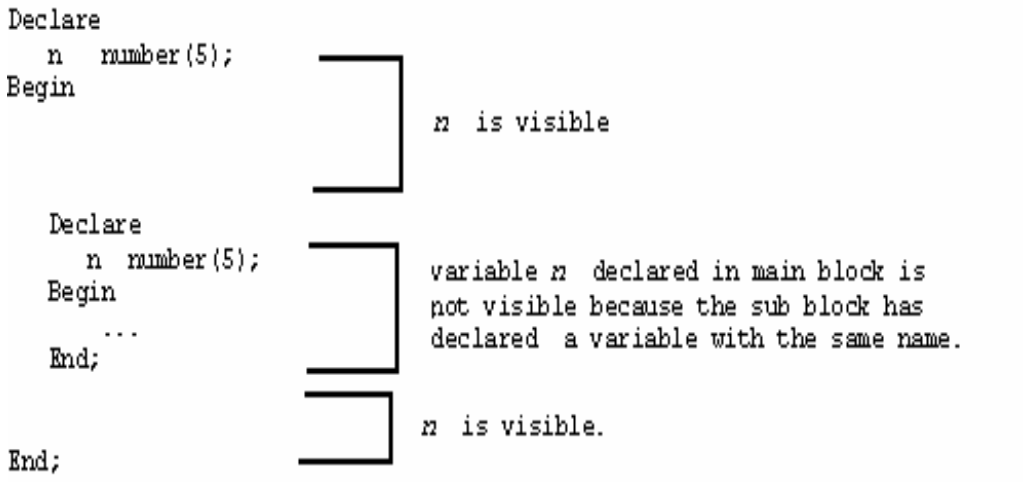


Variable NUM1 can be accessed from the point of declaration to the end of the outer block.

Variable NUM2 can be accessed from the point of declaration to the end of inner block.

Example2:

The following example will illustrate the difference between scope and visibility.



Variable N has scope throughout the outer block, but it is not visible in the inner block as we declared another variable (N) with the same name as the variable in outer block (N). So variable N that is declared in outer block is not visible in inner block instead N that is declared in the inner block is accessed from inner block.

It is possible to access variable N that is declared in the outer block although it is hidden by using label of the outer block.

Labeling the block

A label can be used to name a block. Label is placed within **<<** and **>>** just before the beginning of the block.

The following example shows how to assign a label to a block.

```
<<mainblock>>
declare
     ...
Begin
     ...
end;
```

Label of the block can be used to access hidden objects of the block. For instance, in the previous section, we have seen two blocks declaring a variable with the same name (N), and variable N of the main block is invisible and inaccessible throughout the second block.

To access the hidden variable of main block from inner block, main block may be given a label and the label may be used to refer to hidden objects from inner block as shown below:

```
<<mainblock>>
Declare
   n number(5);
Begin

   <<nestedblock>>
   Declare
      n number(5);
   Begin
        ...

      n := 20;  -- stores  20 into N of nested block.

      /* The following stores 50 into N of main block */

      mainblock.n := 50;

   End;

End;
```

In order to access a hidden variable, we have to prefix the variable with the name of the block. In the above example to access variable N that is declared in main block but hidden in the inner block as another variable is declared with the same name, we use MAINBLOCK.N, where MAINBLOCK is the label given to the block and N is the name of the variable that we want to access.

Assignment Operator (:=)

Assignment operator allows a value to be stored in a variable.

```
variable  :=   expression;
```

The following are examples of assignment operator:

```
count    := 1;
name     := 'Srikanth';
hra      := bs * 0.05;
surname  := substr('P.Srikanth',1,3);
```

In expression you can use the following arithmetic operators.

Operator	What it does?
+	Addition
-	Subtraction
/	Division
*	Multiplication
**	Exponentiation

Available functions

Most of the functions available in SQL are also available in PL/SQL.

The functions that are NOT available in procedural statements are:

- DECODE
- AVG, COUNT, GROUPING, MIN, MAX, SUM, STDDEV, and VARIANCE

However, these functions can be used with SQL commands and those SQL commands may be used in PL/SQL.

Another PL/SQL block

The following is a PL/SQL block to change the course fee of VBNET course with the greatest of average course fee of all courses and Oracle course fee.

```
declare
   v_avgfee courses.fee%type;
   v_orafee courses.fee%type;

begin

    -- get average fee of all courses

    select  avg(fee) into v_avgfee
    from  courses;

    -- get fee of Oracle

    select  fee into v_orafee
    from  courses
    where  ccode = 'ora';

    -- update VB fee with the greatest of these two

    update courses set fee = greatest( v_avgfee, v_orafee)
    where  ccode = 'vbnet';

    -- commit changes

    commit;

end;
/
```

Displaying output from PL/SQL

In order to display output from a PL/SQL block, we have to use DBMS_OUTPUT package. A package is a collection of procedures and functions. We will see more about package in later chapter.

DBMS_OUTPUT is a package that comes along with Oracle database and used to display data onto screen from a PL/SQL block. The following are the procedures available in DBMS_OUTPUT package.

PUT and PUT_LINE procedures

Both these procedures are used to display a NUMBER, VARCHAR2 or DATE type value. PUT allows you to put multiple pieces that make up a line. PUT_LINE puts the given data followed by end-of-line character.

In order to see the output sent using these procedures, the following must satisfy:

- The program unit from where they are called must be completed
- **SERVEROUPUT** variable of SQL*PLUS must be set to ON.

Note: *You must set SERVEROUPUT variable on (shown below) in SQL*PLUS before you run this PL/SQL program. This is to be done in each new session of SQL*PLUS.*

.

SQL> SET SERVEROUPUT ON

The following program is used to display the difference between average duration of Oracle batches and the duration of Oracle course.

```
declare
   v_amtcltd  number(6);
   v_totamt   number(6);
   v_orafee   courses.fee%type;
   v_studcnt  number(3);
   v_diff     number(6);
begin

   -- get total amount collected from Oracle students

   select  sum(amount) into v_amtcltd
   from payments
   where rollno in
       ( select rollno from students
         where  bcode in
         (
           select bcode
           from  batches
           where ccode = 'ora'
         )
       );


   -- get total amount to be collected from Oracle students


   -- first get course fee of Oracle

   select  fee into v_orafee
   from  courses
   where ccode = 'ora';

   -- get no. of students in Oracle batches

   select count(*) into v_studcnt
   from  students
```

```
   where  bcode in
      (
        select bcode from batches
        where  ccode  = 'ora'
      );


  /* calculate difference between total amount to be collection
       and the amount colleted so far */

   v_diff := v_orafee * v_studcnt - v_amtcltd;

   dbms_output.put_line('Oracle arrears : ' || v_diff );

end;
/
```

The following are the important steps in the above program:

1. Getting total amount collected from Oracle students. This can be done by taking rows related to payments of Oracle students.
2. Finding out the number of Oracle students.
3. Finding out the course fee for Oracle.
4. Then multiplying the number of Oracle students with Oracle course fee. This will yield the total amount to be collected from Oracle students.
5. Subtract total amount collected from the figure obtained from previous step to get total amount yet to be paid by Oracle students. Display this value.

Summary

In this chapter, we have seen what is PL/SQL, its features, and advantages. PL/SQL Engine is the component of Oracle that executes PL/SQL blocks. PL/SQL programs are written as PL/SQL blocks where each block contains Declarative Part, Executable Part, and Exception Handler part.

PL/SQL allows programmers to retrieve the data from database using SQL and then process the data with its conditional statements and looping structures. PL/SQL is a superset of SQL.

Exercises

1. In _____ part of PL/SQL block errors are handled.
2. ___, ____ and _____ are the valid values for BOOLEAN data type.
3. The part of Oracle Server that executes SQL commands is called as _____.
4. _____ is an example of Oracle tool that contains PL/SQL engine.
5. _____ is the operator for exponentiation.
6. _____ is used for commenting a single line.
7. Write PL/SQL block to change the DURATION of courses C++ to the duration of Java course.
8. Insert a new row into COURSE_FACULTY table with the following details:
   Course name is Oracle8i, Faculty name is Kevin Loney, and grade is B.