Database Programming

PL/SQL Concepts

Sagara Samarawickrama

2024W

PL/SQL Concepts

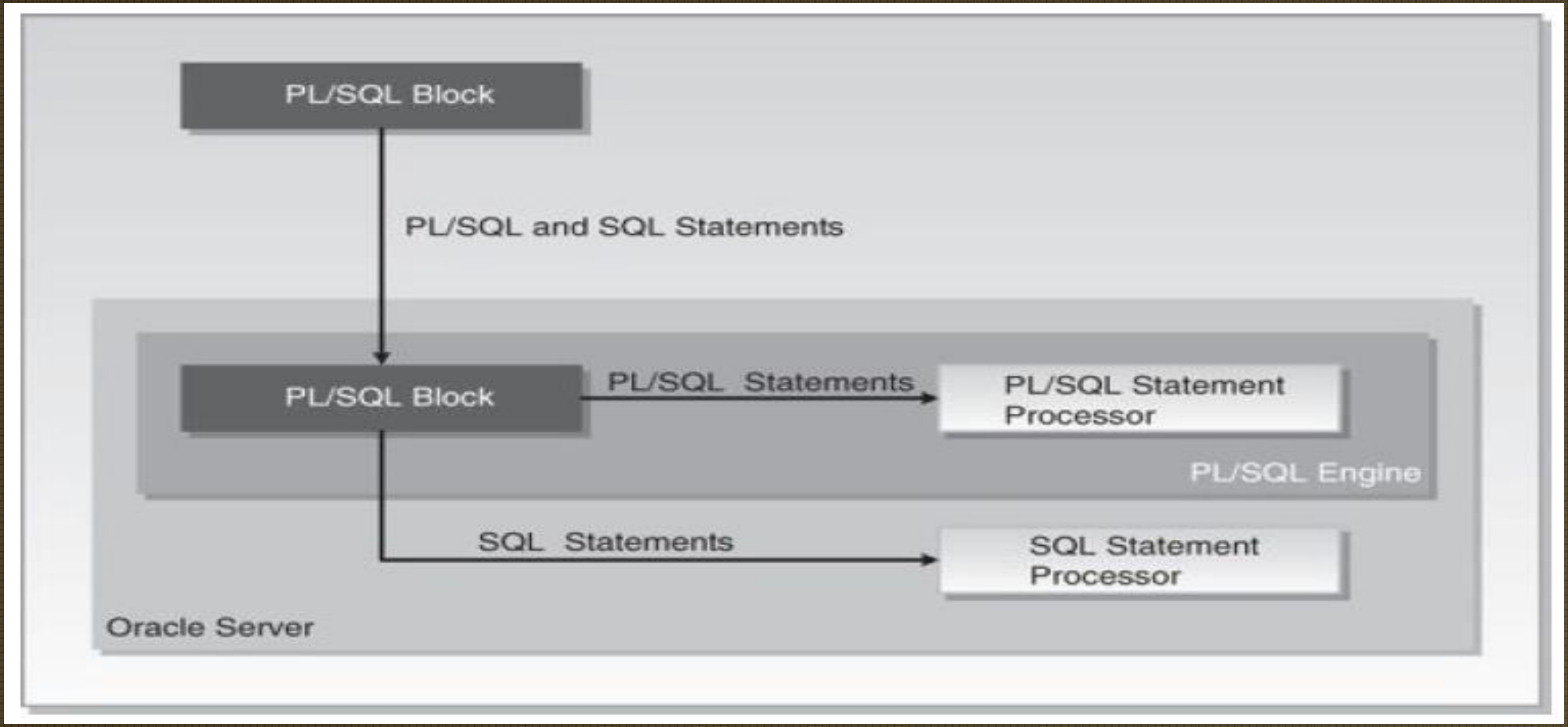
PL/SQL stands for “Procedural Language Extension to SQL.” Because of its tight integration with SQL, PL/SQL supports the great majority of the SQL features, such as SQL data manipulation, data types, operators, functions, and transaction control statements. As an extension to SQL, PL/SQL combines SQL with programming structures

and subroutines available in any high-level language.

PL/SQL Architecture:

While PL/SQL is just like any other programming language, its main distinction is that it is not a stand-alone programming language. Rather, PL/SQL is a part of the Oracle RDBMS as well as various Oracle development tools such as Oracle Application Express (APEX) and Oracle Forms and Reports

PL/SQL Concepts



When the PL/SQL engine is located on the server, the whole PL/SQL block is passed to the PL/SQL engine on the Oracle server. The PL/SQL engine processes the block according to the scheme depicted in the above figure

PL/SQL Concepts

Using PL/SQL has several advantages.

For example, when you issue a SELECT statement in SQL*Plus or SQL Developer against the STUDENT table, it retrieves a list of students. The SELECT statement you issued at the client computer is sent to the database server to be executed. The results of this execution are then returned to the client. In turn, rows are displayed on your client machine.

Now, assume that you need to issue multiple SELECT statements. Each SELECT statement is a request against the database and is sent to the Oracle server. The results of each SELECT statement are sent back to the client. Each time a SELECT statement is executed, network traffic is generated. Hence, multiple SELECT statements will result in multiple round-trip transmissions, adding significantly to the network traffic.

PL/SQL Concepts

When these SELECT statements are combined into a PL/SQL program they are sent to the server as a single unit. The SELECT statements in this PL/SQL program are executed at the server. The server sends the results of these SELECT statements back to the client, also as a single unit. Therefore, a PL/SQL program encompassing multiple SELECT

statements can be executed at the server and have all of the results returned to the client in the same round trip. This is obviously a more efficient process than having each SELECT statement execute independently.

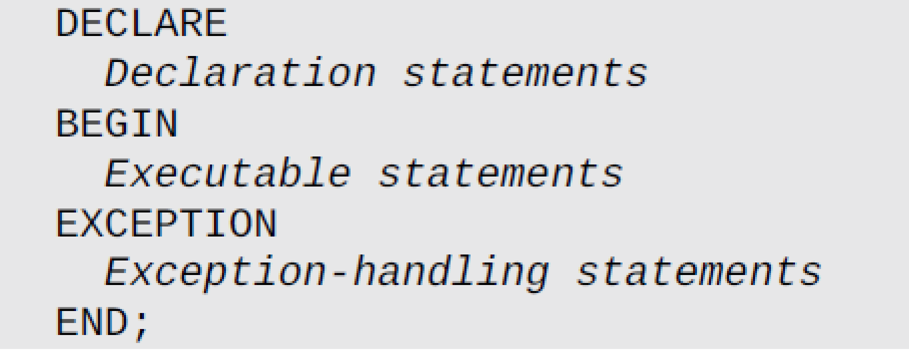
PL/SQL Block Structure

A block is the most basic unit in PL/SQL. All PL/SQL programs are combined into blocks. These blocks can also be nested within one another. Usually, PL/SQL blocks combine statements that represent a single logical task

PL/SQL Concepts

PL/SQL blocks can be divided into two groups: named and anonymous. Named PL/SQL blocks are used when creating subroutines. These subroutines, which include procedures, functions, and packages, can be stored in the database and referenced by their names later

PL/SQL blocks contain three sections: a declaration section, an executable section, and an exception-handling section. The executable section is the only mandatory section of the block; both the declaration and exception-handling sections are optional



```
DECLARE
  Declaration statements
BEGIN
  Executable statements
EXCEPTION
  Exception-handling statements
END;
```

PL/SQL Concepts

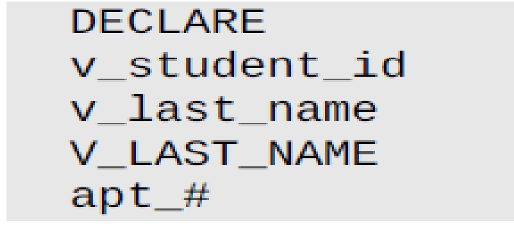
PL/SQL Language Components

Character Types

The PL/SQL engine accepts four types of characters: letters, digits, symbols (*, +, –, =,and so on), and white space. When elements from one or more of these character types are joined together, they create a lexical unit (lexical units can be a combination of character types). The lexical units are the words of the PL/SQL language

PL/SQL Variables

Variables may be used to hold a temporary value. Variables may also be known as identifiers. There are some restrictions that you need to be familiar with. Specifically, variables must begin with a letter and may be up to 30 characters long. Consider the following example, which contains a list of valid identifiers:

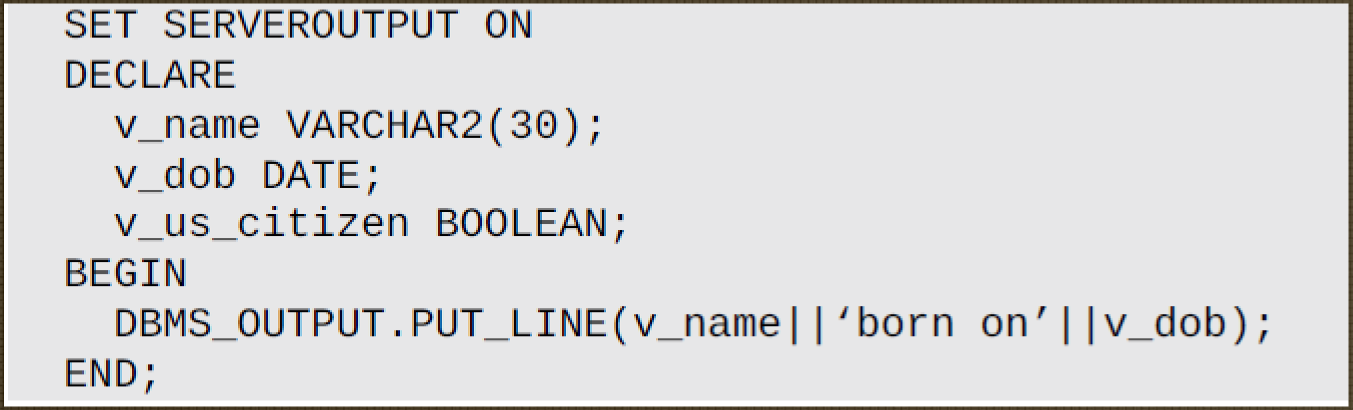


```
DECLARE
v_student_id
v_last_name
V_LAST_NAME
apt_#
```

PL/SQL Concepts

PL/SQL Language Components

What is the output of the following program ?



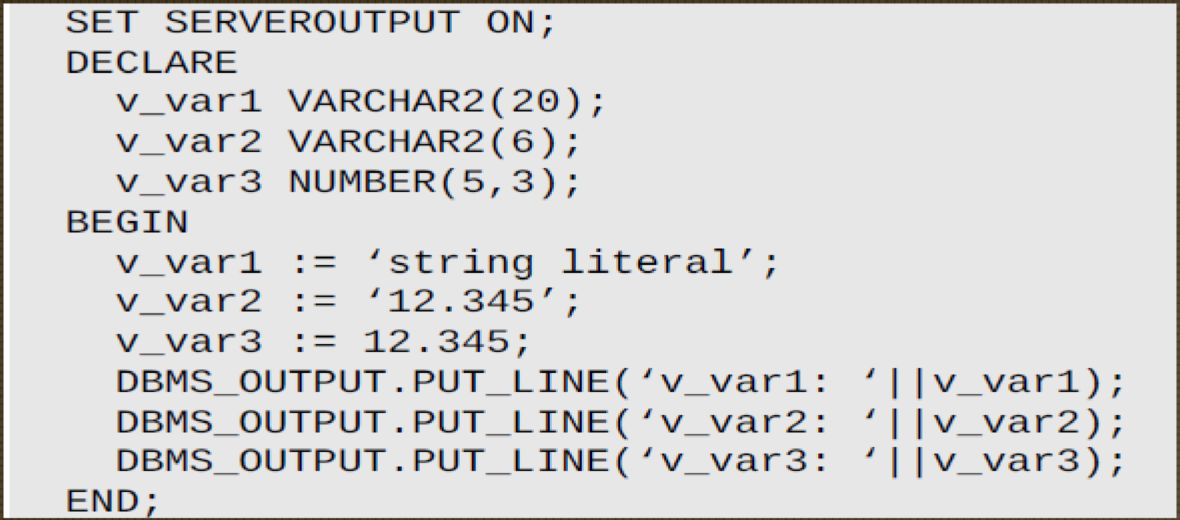
```
SET SERVEROUTPUT ON
DECLARE
  v_name VARCHAR2(30);
  v_dob DATE;
  v_us_citizen BOOLEAN;
BEGIN
  DBMS_OUTPUT.PUT_LINE(v_name||‘born on’||v_dob);
END;
```

Reserved words are ones that PL/SQL saves for its own use (e.g., BEGIN, END, and SELECT). You cannot use reserved words for names of variables, literals, or user-defined exceptions.

Identifiers in PL/SQL

Take a look at the use of identifiers in the following example:

PL/SQL Concepts



```
SET SERVEROUTPUT ON;
DECLARE
   v_var1 VARCHAR2(20);
   v_var2 VARCHAR2(6);
   v_var3 NUMBER(5,3);
BEGIN
   v_var1 := ‘string literal’;
   v_var2 := ‘12.345’;
   v_var3 := 12.345;
   DBMS_OUTPUT.PUT_LINE(‘v_var1: ‘||v_var1);
   DBMS_OUTPUT.PUT_LINE(‘v_var2: ‘||v_var2);
   DBMS_OUTPUT.PUT_LINE(‘v_var3: ‘||v_var3);
END;
```

Most Common Data Types

VARCHAR2 (maximum_length)

Stores variable-length character data.Takes a required parameter that specifies a maximum length up to 32,767 bytes, with the Extended Data Types parameter enabled. Otherwise, the maximum length is 4000

PL/SQL Concepts

CHAR[(maximum_length)]

Stores fixed-length (blank-padded if necessary) character data.Takes an optional parameter that specifies a maximum length up to 32,767 bytes. Does not use a constant or variable to specify the maximum length; an integer literal must be used. If maximum length is not specified, it defaults to 1

NUMBER[(precision, scale)]

Stores fixed or floating-point numbers of virtually any size.The precision is the total number of digits. The scale determines where rounding occurs.

DATE

Stores fixed-length date values. Valid dates for DATE variables include January 1, 4712 BC, to December 31,AD 9999.

PL/SQL Concepts

LONG

Stores variable-length character strings.The LONG data type is like the VARCHAR2 data type, except that the maximum length of a LONG value is 2 gigabytes.

Labels and Nested Blocks

Labels can be added to a block to improve readability and to qualify the names of elements that exist under the same name in nested blocks. The name of the block must precede the first line of executable code (either BEGIN or DECLARE) as follows:

```
set serveroutput on
  << find_stu_num >>
  BEGIN
    DBMS_OUTPUT.PUT_LINE(‘The procedure
           find_stu_num has been executed.’);
  END find_stu_num;
```

Blocks can be nested in the main section or in an exception handler. A nested block is a block that is placed fully within another block.

Q
&
A