Oracle PL/SQL is an extension of SQL language, designed for seamless processing of SQL statements enhancing the security, portability, and robustness of the database. This PL/SQL online programming course explains some important aspect of PL SQL language like block structure, data types, packages, triggers, exception handling, etc.

**What Is PL/SQL? Introduction & Architecture**

## What is PL/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. Full form of PL/SQL is "Procedural Language extensions to SQL".

It combines the data manipulation power of SQL with the processing power of procedural language to create super powerful SQL queries.

PL/SQL means instructing the compiler 'what to do' through SQL and 'how to do' through its procedural way.

Similar to other database languages, it gives more control to the programmers by the use of loops, conditions and object-oriented concepts.

In this tutorial, you will learn-

* [What is PL/SQL?](https://www.guru99.com/introduction-pl-sql.html#1)
* [Architecture of PL/SQL](https://www.guru99.com/introduction-pl-sql.html#2)
* [Advantage of Using PL/SQL](https://www.guru99.com/introduction-pl-sql.html#3)

## Architecture of PL/SQL

The PL/SQL architecture mainly consists of following three components:

1. PL/SQL block
2. PL/SQL Engine
3. Database Server

### PL/SQL block:

* This is the component which has the actual PL/SQL code.
* This consists of different sections to divide the code logically (declarative section for declaring purpose, execution section for processing statements, exception handling section for handling errors)
* It also contains the SQL instruction that used to interact with the database server.
* All the PL/SQL units are treated as PL/SQL blocks, and this is the starting stage of the architecture which serves as the primary input.
* Following are the different type of PL/SQL units.
  + Anonymous Block
  + Function
  + Library
  + Procedure
  + Package Body
  + Package Specification
  + Trigger
  + Type
  + Type Body

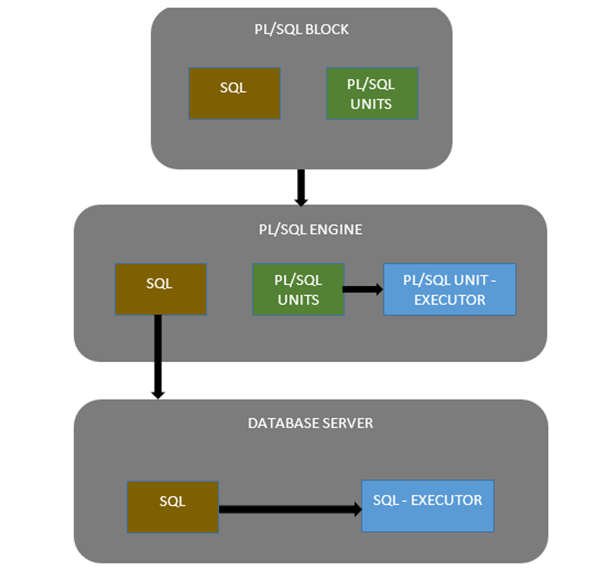
### PL/SQL Engine

* PL/SQL engine is the component where the actual processing of the codes takes place.
* PL/SQL engine separates PL/SQL units and SQL part in the input (as shown in the image below).
* The separated PL/SQL units will be handled by the PL/SQL engine itself.
* The SQL part will be sent to database server where the actual interaction with database takes place.
* It can be installed in both database server and in the application server.

### Database Server:

* This is the most important component of Pl/SQL unit which stores the data.
* The PL/SQL engine uses the SQL from PL/SQL units to interact with the database server.
* It consists of SQL executor which parses the input SQL statements and execute the same.

Below is the pictorial representation of Architecture of PL/SQL.

[](https://www.guru99.com/images/PL-SQL/110215_0504_Introductio1.png)

PL/SQL Architecture Diagram

## Advantage of Using PL/SQL

1. Better performance, as SQL is executed in bulk rather than a single statement
2. High Productivity
3. Tight integration with SQL
4. Full Portability
5. Tight Security
6. Support Object Oriented Programming concepts.

# SQL Vs PL/SQL Vs T-SQL: Key Differences

QL is the standard language to query a database.

PL SQL basically stands for "Procedural Language extensions to SQL." This is the extension of Structured Query Language (SQL) that is used in Oracle.

T-SQL basically stands for " Transact-SQL." This is the extension of Structured Query Language (SQL) that is used in Microsoft.

In this tutorial, we will learn-

* [Difference between SQL and PL/SQL](https://www.guru99.com/sql-vs-pl-sql.html#1)
* [Difference Between T-SQL and PL-SQL](https://www.guru99.com/sql-vs-pl-sql.html#2)
* [Difference between SQL and T-SQL](https://www.guru99.com/sql-vs-pl-sql.html#3)

## Difference between SQL and PL/SQL

|  |  |
| --- | --- |
| **SQL** | **PL/SQL** |
| * SQL is a single query that is used to perform DML and DDL operations. | * PL/SQL is a block of codes that used to write the entire program blocks/ procedure/ function, etc. |
| * It is declarative, that defines what need to be done, rather than how things need to be done. | * PL/SQL is procedural that defines how the things needs to be done. |
| * Execute as a single statement. | * Execute as a whole block. |
| * Mainly used to manipulate data. | * Mainly used to create an application. |
| * Interaction with a Database server. | * No interaction with the database server. |
| * Cannot contain PL/SQL code in it. | * It is an extension of SQL, so that it can contain SQL inside it. |

## Difference Between T-SQL and PL-SQL

|  |  |
| --- | --- |
| **T-SQL** | **PL-SQL** |
| * T-SQL is a Microsoft product. | * PL-SQL is developed by Oracle. |
| * Full Form of TL SQL is Transact Structure Query language. | * Full Form of PL SQL is Procedural Language Structural Query Language. |
| * T-SQL gives a high degree of control to programmers. | * It is a natural programming language that blends easily with the SQL |
| * T-SQL performs best with Microsoft SQL server | * PL-SQL performs best with Oracle database server. |
| * It is easy and simple to understand. | * PL-SQL is complex to understand. |
| * T-SQL allows inserting multiples rows into a table using the BULK INSERT statement. | * PL/SQL supports oops concepts like data encapsulation, function overloading, and information hiding. |
| * SELECT INTO statement used in T-SQL | * INSERT INTO statement must be used in PL/SQL |
| * In T-SQL NOT EXISTS clause used along with SELECT statements. | * In PL/SQL, there is a MINUS operator, which could be used with SELECT statements |

## Difference between SQL and T-SQL

|  |  |
| --- | --- |
| **SQL** | **T-SQL** |
| * SQL is a programming language which focuses on managing relational databases. | * T-SQL is a procedural extension used by SQL Server. |
| * This is used for controlling and manipulating data where large amounts of information are stored about products, clients, etc. | * T-SQL has some features that are not available in SQL. Like procedural programming elements and a local variable to provide more flexible control of how the application flows. |
| * SQL queries submitted individually to the database server. | * T-SQL writes a program in such a way that all commands are submitted to the server in a single go |
| * The syntax was formalized for many commands; some of these are SELECT, INSERT, UPDATE, DELETE, CREATE, and DROP. | * It also includes special functions like the converted date () and some other functions which are not part of the regular SQL. |

# PL/ SQL Block: STRUCTURE, Syntax, ANONYMOUS Example

## What is PL/SQL block?

In PL/SQL, the code is not executed in single line format, but it is always executed by grouping the code into a single element called Blocks. In this tutorial, you are going to learn about these blocks.

Blocks contain both PL/SQL as well as SQL instruction. All these instruction will be executed as a whole rather than executing a single instruction at a time.

you will learn:

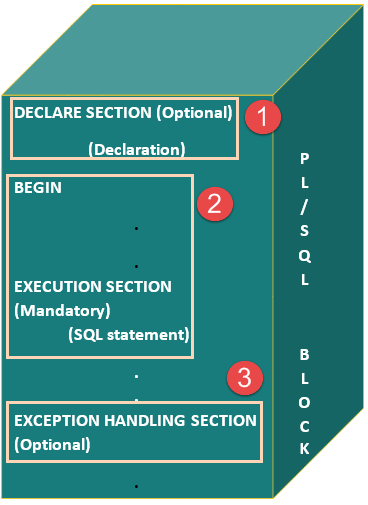
* [What is PL/SQL block?](https://www.guru99.com/blocks-pl-sql.html#1)
* [Block Structure](https://www.guru99.com/blocks-pl-sql.html#2)
* [PL/SQL Block Syntax](https://www.guru99.com/blocks-pl-sql.html#4)
* [Types of PL/SQL block](https://www.guru99.com/blocks-pl-sql.html#3)

## Block Structure

PL/SQL blocks have a pre-defined structure in which the code is to be grouped. Below are different sections of PL/SQL blocks.

1. Declaration section
2. Execution section
3. Exception-Handling section

The below picture illustrates the different PL/SQL block and their section order.

[](https://www.guru99.com/images/PL-SQL/110215_0632_BlocksinPLS1.png)

### Declaration Section

This is the first section of the PL/SQL blocks. This section is an optional part. This is the section in which the declaration of variables, cursors, exceptions, subprograms, pragma instructions and collections that are needed in the block will be declared. Below are few more characteristics of this part.

* This particular section is optional and can be skipped if no declarations are needed.
* This should be the first section in a PL/SQL block, if present.
* This section starts with the keyword 'DECLARE' for triggers and anonymous block. For other subprograms, this keyword will not be present. Instead, the part after the subprogram name definition marks the declaration section.
* This section should always be followed by execution section.

### Execution Section

Execution part is the main and mandatory part which actually executes the code that is written inside it. Since the PL/SQL expects the executable statements from this block this cannot be an empty block, i.e., it should have at least one valid executable code line in it. Below are few more characteristics of this part.

* This can contain both PL/SQL code and SQL code.
* This can contain one or many blocks inside it as a nested block.
* This section starts with the keyword 'BEGIN'.
* This section should be followed either by 'END' or Exception-Handling section (if present)

### Exception-Handling Section:

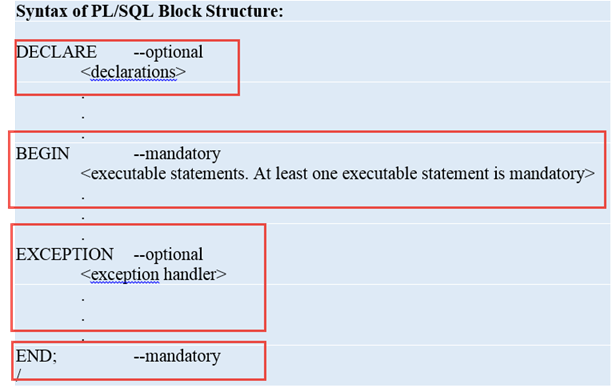
The exception is unavoidable in the program which occurs at run-time and to handle this Oracle has provided an Exception-handling section in blocks. This section can also contain PL/SQL statements. This is an optional section of the PL/SQL blocks.

* This is the section where the exception raised in the execution block is handled.
* This section is the last part of the PL/SQL block.
* Control from this section can never return to the execution block.
* This section starts with the keyword 'EXCEPTION'.
* This section should always be followed by the keyword 'END'.

The Keyword 'END' marks the end of PL/SQL block.

## PL/SQL Block Syntax

Below is the syntax of the PL/SQL block structure.

[](https://www.guru99.com/images/PL-SQL/110215_0632_BlocksinPLS2.png)

DECLARE --optional

<declarations>

BEGIN --mandatory

<executable statements. At least one executable statement is mandatory>

EXCEPTION --optional

<exception handles>

END; --mandatory

/

**Note:** A block should always be followed by '/' which sends the information to the compiler about the end of the block.

## Types of PL/SQL block

PL/SQL blocks are of mainly two types.

1. Anonymous blocks
2. Named Blocks

### Anonymous blocks:

Anonymous blocks are PL/SQL blocks which do not have any names assigned to them. They need to be created and used in the same session because they will not be stored in the server as database objects.

Since they need not store in the database, they need no compilation steps. They are written and executed directly, and compilation and execution happen in a single process.

Below are few more characteristics of Anonymous blocks.

* These blocks don't have any reference name specified for them.
* These blocks start with the keyword 'DECLARE' or 'BEGIN'.
* Since these blocks do not have any reference name, these cannot be stored for later purpose. They shall be created and executed in the same session.
* They can call the other named blocks, but call to anonymous block is not possible as it is not having any reference.
* It can have nested block in it which can be named or anonymous. It can also be nested in any blocks.
* These blocks can have all three sections of the block, in which execution section is mandatory, the other two sections are optional.

### Named blocks:

Named blocks have a specific and unique name for them. They are stored as the database objects in the server. Since they are available as database objects, they can be referred to or used as long as it is present on the server. The compilation process for named blocks happens separately while creating them as a database objects.

Below are few more characteristics of Named blocks.

* These blocks can be called from other blocks.
* The block structure is same as an anonymous block, except it will never start with the keyword 'DECLARE'. Instead, it will start with the keyword 'CREATE' which instruct the compiler to create it as a database object.
* These blocks can be nested within other blocks. It can also contain nested blocks.
* Named blocks are basically of two types:

1. Procedure
2. Function

We will learn more about these named blocks in "Procedure" and "Function" topics in later tutorial.

## Summary

After this tutorial, you should be aware of PL/SQL blocks and its types, different sections of blocks and their usages. The detailed description of the named PL/SQL blocks will be covered in the later tutorial.

# PL/SQL First Program: Hello World Example

we will introduce SQL\* Plus and learn how to connect it to the database.

After connection, we are also going to see how to write our first program "Hello World" in PL/SQL.

In this tutorial - you will learn.

* [What is SQL\* Plus?](https://www.guru99.com/pl-sql-first-program-helloworld.html#1)
* [Connecting to Database](https://www.guru99.com/pl-sql-first-program-helloworld.html#2)
* [How to write a simple program using PL/SQL](https://www.guru99.com/pl-sql-first-program-helloworld.html#3)
* [Declaring and usage of variables in the program](https://www.guru99.com/pl-sql-first-program-helloworld.html#4)
* [Comments in PL/SQL](https://www.guru99.com/pl-sql-first-program-helloworld.html#5)

## What is SQL\* Plus?

SQL\* Plus is an interactive and batch query tool that is installed with every Oracle installation. It can be found at Start > Programs > Oracle-OraHomeName > Application Development > SQL Plus. Alternatively, you can also download it from the Oracle Technology Network (OTN)

It has a command line user interface, Windows GUI, and web-based user interface.

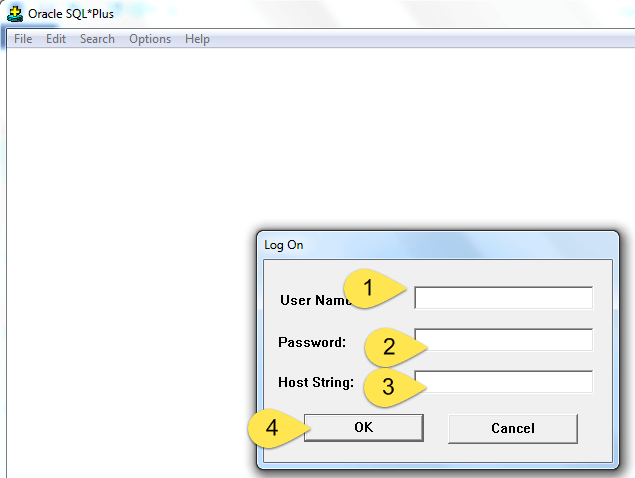
It allows the user to connect to the database and execute PL/SQL commands.

## Connecting to Database

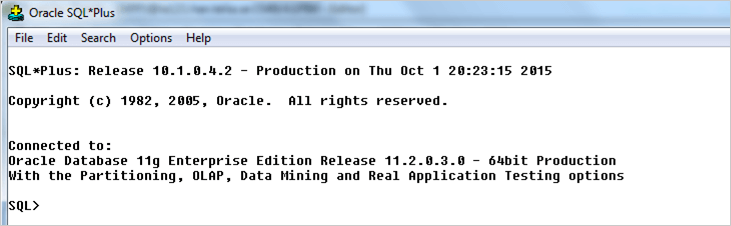
In this section, we are going to learn how to connect to SQL\* Plus in Windows GUI. When we open SQL\* Plus, it will prompt for the connection details as shown below.

### Connection Details:

* Username: <user name of the database>
* Password: <password for that user>
* Host String: <host details along with the port number and SID of the database>

[](https://www.guru99.com/images/PL-SQL/110215_0652_PLSQLFirstP1.png)

* After the successful connection, the[SQL](https://www.guru99.com/sql.html)plus will appear as shown below

[](https://www.guru99.com/images/PL-SQL/110215_0652_PLSQLFirstP2.png)

* We need to execute "set serveroutput on" if we need to see the output of the code.
* Now we are ready to work with the SQL\* Plus tool.

## How to write a simple program using PL/SQL

In this section, we are going to write a simple program for printing "Hello World" using "Anonymous block".

BEGIN

dbms\_output.put\_line (‘Hello World..');

END:

/

**Output:**

Hello World...

**Code Explanation:**

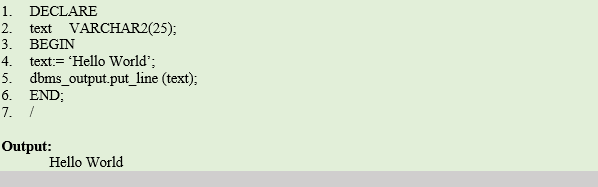
* **Code line 2**: Prints the message "Hello World. . ."
* The below screenshot explains how to enter the code in SQL\* Plus.

**Note:** A block should be always followed by '/' which sends the information to the compiler about the end of the block. Till the compiler encounters '/', it will not consider the block is completed, and it will not execute it.

[](https://www.guru99.com/images/PL-SQL/110215_0652_PLSQLFirstP4.png)

## Declaring and usage of variables in the program

Here we are going to print the "Hello World" using the variables.

[](https://www.guru99.com/images/PL-SQL/110215_0652_PLSQLFirstP5.png)

DECLARE

text VARCHAR2(25);

BEGIN

text:= ‘Hello World’;

dbms\_output.put\_line (text);

END:

/

**Output:**

Hello World

* **Code line 2**: Declaring a variable "text" of a VARCHAR2 type with size 25
* **Code line 4**: Assigning the value "Hello World" to the variable "text".
* **Code line 5**: Printing the value of the variable "text".

## Comments in PL/SQL

Commenting code simply instructs the compiler to ignore that particular code from executing.

Comment can be used in the program to increase the readability of the program. In PL/SQL codes can be commented in two ways.

* Using '--' in the beginning of the line to comment that particular line.
* Using '/\*…….\*/' we can use multiple lines. The symbol '/\*' marks the starting of the comment and the symbol '\*/' marks the end of the comment. The code between these two symbols will be treated as comments by the compiler.

**Example**: In this example, we are going to print 'Hello World' and we are also going to see how the commented lines behave in the code

[](https://www.guru99.com/images/PL-SQL/110215_0652_PLSQLFirstP6.png)

BEGIN

--single line comment

dbms output.put line (' Hello World ’);

/\*Multi line commenting begins

Multi line commenting ends \*/

END;

/

**Output:**

Hello World

**Code Explanation:**

* **Code line 2**: Single line comment and compiler ignored this line from execution.
* **Code line 3**: Printing the value "Hello World."
* **Code line 4**: Multiline commenting starts with '/\*'
* **Code line 5**: Multiline commenting ends with '\*/'

## Summary

In this tutorial, you have learned about SQL\* Plus and Connection establishment to SQL\* Plus. You have also learned about how to write the simple program and how to use a variable in them. In our upcoming chapters, we will learn more about different functionalities that can be implemented in the PL SQL program.

# Oracle PL/SQL Data Types: Character, Number, Boolean, Date, LOB

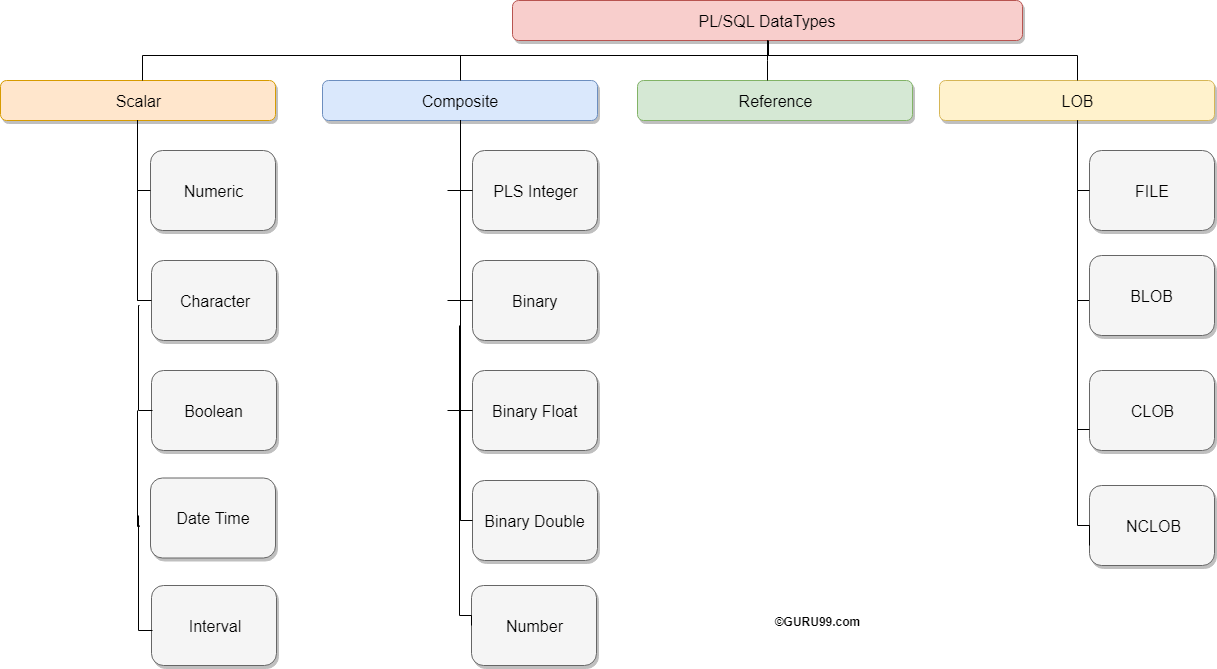
## What is PL/SQL Datatypes?

A data type is associated with the specific storage format and range constraints. In Oracle, each value or constant is assigned with a data type.

Basically, it defines how the data is stored, handled and treated by Oracle during the data storage and processing.

The main difference between PL/SQL and[SQL](https://www.guru99.com/sql.html)data types is, SQL data type are limited to table column while the PL/SQL data types are used in the PL/SQL blocks. More on this later in the tutorial.

Following is the diagram of different Data Types in PL/SQL

[[](https://www.guru99.com/images/1/plsql_datatype.png)](https://www.guru99.com/images/1/plsql_datatype.png)

In this tutorial, you will learn-

* [CHARACTER Data Type](https://www.guru99.com/pl-sql-data-types.html#1)
* [NUMBER Data Type](https://www.guru99.com/pl-sql-data-types.html#2)
* [BOOLEAN Data Type](https://www.guru99.com/pl-sql-data-types.html#3)
* [DATE Data Type](https://www.guru99.com/pl-sql-data-types.html#4)
* [LOB Data Type](https://www.guru99.com/pl-sql-data-types.html#5)

## CHARACTER Data Type:

This data type basically stores alphanumeric characters in string format.

The literal values should always be enclosed in single quotes while assigning them to CHARACTER data type.

This character data type is further classified as follows:

* CHAR Data type (fixed string size)
* VARCHAR2 Data type (variable string size)
* VARCHAR Data type
* NCHAR (native fixed string size)
* NVARCHAR2 (native variable string size)
* LONG and LONG RAW

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Description** | **Syntax** |
| **CHAR** | This data type stores the string value, and the size of the string is fixed at the time of declaring the variable.   * Oracle would be blank-padded the variable if the variable didn't occupy the entire size that has been declared for it, Hence Oracle will allocate the memory for declared size even if the variable didn't occupy it fully. * The size restriction for this data type is 1-2000 bytes. * CHAR data type is more appropriate to use where ever fixed the size of data will be handled. | grade CHAR;  manager CHAR (10):= 'guru99';  **Syntax Explanation:**   * The first declaration statement declared the variable 'grade' of CHAR data type with the maximum size of 1 byte (default value). * The second declaration statement declared the variable 'manager' of CHAR data type with the maximum size of 10 and assigned the value 'guru99' which is of 6 bytes. Oracle will allocate the memory of 10 bytes rather than 6 bytes in this case. |
| **VARCHAR2** | This data type stores the string, but the length of the string is not fixed.    * The size restriction for this data type is 1-4000 bytes for table column size and 1-32767 bytes for variables. * The size is defined for each variable at the time of variable declaration. * But Oracle will allocate memory only after the variable is defined, i.e., Oracle will consider only the actual length of the string that is stored in a variable for memory allocation rather than the size that has been given for a variable in the declaration part. * It is always good to use VARCHAR2 instead of CHAR data type to optimize the memory usage. | manager VARCHAR2(10) := ‘guru99';  **Syntax Explanation:**   * The above declaration statement declared the variable 'manager' of VARCHAR2 data type with the maximum size of 10 and assigned the value 'guru99' which is of 6 bytes. Oracle will allocate memory of only 6 bytes in this case. |
| **VARCHAR** | This is synonymous with the VARCHAR2 data type.    * It is always a good practice to use VARCHAR2 instead of VARCHAR to avoid behavioral changes. | manager VARCHAR(10) := ‘guru99';  **Syntax Explanation:**   * The above declaration statement declared the variable 'manager' of VARCHAR data type with the maximum size of 10 and assigned the value 'guru99' which is of 6 bytes. Oracle will allocate memory of only 6 bytes in this case. (Similar to VARCHAR2) |
| **NCHAR** | This data type is same as CHAR data type, but the character set will of the national character set.    * This character set can be defined for the session using NLS\_PARAMETERS. * The character set can be either UTF16 or UTF8. * The size restriction is 1-2000 bytes. | native NCHAR(10);  **Syntax Explanation:**   * The above declaration statement declares the variable 'native' of NCHAR data type with the maximum size of 10. * The length of this variable depends upon the (number of lengths) per byte as defined in the character set. |
| **NVARCHAR2** | This data type is same as VARCHAR2 data type, but the character set will be of the national character set.   * This character set can be defined for the session using NLS\_PARAMETERS. * The character set can be either UTF16 or UTF8. * The size restriction is 1-4000 bytes. | Native var NVARCHAR2(10):='guru99';  **Syntax Explanation:**   * The above declaration statement declares the variable 'Native\_var' of NVARCHAR2 data type with the maximum size of 10. |
| **LONG and LONGRAW** | This data type is used to store large text or raw data up to the maximum size of 2GB.   * These are mainly used in the data dictionary. * LONG data type is used to store character set data, while LONG RAW is used to store data in binary format. * LONG RAW data type accepts media objects, images, etc. whereas LONG works only on data that can be stored using character set. | Large\_text LONG;  Large\_raw LONG RAW;  **Syntax Explanation:**   * The above declaration statement declares the variable 'Large\_text' of LONG data type and 'Large\_raw' of LONG RAW data type.   **Note:** Using LONG data type is not recommended by Oracle. Instead, LOB data type should be preferred. |

## NUMBER Data Type:

This data type stores fixed or floating point numbers up to 38 digits of precision. This data type is used to work with fields which will contain only number data. The variable can be declared either with precision and decimal digit details or without this information. Values need not enclose within quotes while assigning for this data type.

A NUMBER(8,2);

B NUMBER(8);

C NUMBER;

**Syntax Explanation:**

* In the above, the first declaration declares the variable 'A' is of number data type with total precision 8 and decimal digits 2.
* The second declaration declares the variable 'B' is of number data type with total precision 8 and no decimal digits.
* The third declaration is the most generic, declares variable 'C' is of number data type with no restriction in precision or decimal places. It can take up to a maximum of 38 digits.

## BOOLEAN Data Type:

This data type stores the logical values. It represents either TRUE or FALSE and mainly used in conditional statements. Values need not enclose within quotes while assigning for this data type.

Var1 BOOLEAN;

**Syntax Explanation:**

* In the above, variable 'Var1' is declared as BOOLEAN data type. The output of the code will be either true or false based on the condition set.

## DATE Data Type:

This data type stores the values in date format, as date, month, and year. Whenever a variable is defined with DATE data type along with the date it can hold time information and by default time information is set to 12:00:00 if not specified. Values need to enclose within quotes while assigning for this data type.

The standard Oracle time format for input and output is 'DD-MON-YY' and it is again set at NLS\_PARAMETERS (NLS\_DATE\_FORMAT) at the session level.

newyear DATE:='01-JAN-2015';

current\_date DATE:=SYSDATE;

**Syntax Explanation:**

* In the above, variable 'newyear' is declared as DATE data type and assigned the value of Jan 1st, 2015 date.
* The second declaration declares the variable current\_date as DATE data type and assigned the value with current system date.
* Both these variable holds the time information.

## LOB Data Type:

This data type is mainly used to store and manipulate large blocks of unstructured data's like images, multimedia files, etc. Oracle prefers LOB instead of the a LONG data type as it is more flexible than the LONG data type. The below are the few main advantage of LOB over LONG data type.

* The number of column in a table with LONG data type is limited to 1, whereas a table has no restriction on a number of columns with LOB data type.
* The data interface tool accepts LOB data type of the table during data replication, but it omits LONG column of the table. These LONG columns need to be replicated manually.
* The size of the LONG column is 2GB, whereas LOB can store up to 128 TB.
* Oracle is constantly improvising the LOB data type in each of their releases according to the modern requirement, whereas LONG data type is constant and not getting many updates.

So, it is always good to use LOB data type instead of the LONG data type. Following are the different LOB data types. They can store up to the size of 128 terabytes.

1. BLOB
2. CLOB and NCLOB
3. BFILE

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Description** | **Syntax** |
| **BLOB** | This data type stores the LOB data in the binary file format up to the maximum size of 128 TB. This doesn't store data based on the character set details, so that it can store the unstructured data such as multimedia objects, images, etc. | Binary\_data BLOB;  **Syntax Explanation:**   * In the above, variable 'Binary\_data' is declared as a BLOB. |
| **CLOB and NCLOB** | CLOB data type stores the LOB data into the character set, whereas NCLOB stores the data in the native character set. Since these data types use character set based storage, these cannot store the data like multimedia, images, etc. that cannot be put into a character string. The maximum size of these data types is 128 TB. | Charac\_data CLOB;  **Syntax Explanation:**   * In the above, variable 'Charac\_data' is declared as CLOB data type. |
| **BFILE** | * BFILE are the data types that stored the unstructured binary format data outside the database as an operating-system file. * The size of BFILE is to a limited operating system, and they are read-only files and can't be modified. |  |

## Summary

We have covered the different simple data types that are available in PL/SQL along with their syntax. We will learn about complex data types in further topics.