PL/SQL Introduction:

PL/SQL is a block structured language that enables developers to combine the power of SQL with procedural statements. All the statements of a block are passed to oracle engine all at once which increases processing speed and decreases the traffic.

Features of PL/SQL:

1.PL/SQL is basically a procedural language, which provides the functionality of decision making, iteration and many more features of procedural programming languages.
2.PL/SQL can execute a number of queries in one block using single command.
3.One can create a PL/SQL unit such as procedures, functions, packages, triggers, and types, which are stored in the database for reuse by applications.
4.PL/SQL provides a feature to handle the exception which occurs in PL/SQL block known as exception handling block.
5.Applications written in PL/SQL are portable to computer hardware or operating system where Oracle is operational.
6.PL/SQL Offers extensive error checking.

Disadvantages of SQL:

•SQL doesn't provide the programmers with a technique of condition checking, looping and branching.
•SQL statements are passed to Oracle engine one at a time which increases traffic and decreases speed.
•SQL has no facility of error checking during manipulation of data.

Differences 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 needs 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.
Cannot contain PL/SQL code in it.	It is an extension of SQL, so it can contain SQL inside it.

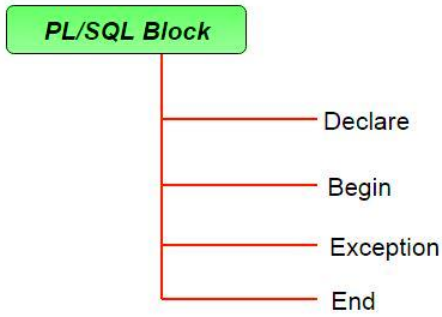
Structure of PL/SQL Block:

PL/SQL extends SQL by adding constructs found in procedural languages, resulting in a structural language that is more powerful than SQL. The basic unit in PL/SQL is a block. All PL/SQL programs are made up of blocks, which can be nested within each other.

Typically, each block performs a logical action in the program. A block has the following structure:

```
DECLARE declaration statements;
BEGIN executable statements
```

```
EXCEPTIONS exception handling statements
END;
```



•Execution section starts with **BEGIN** and ends with **END** keyword.This is a mandatory section and here the program logic is written to perform any task like loops and conditional statements. It supports all DML commands, DDL commands and SQL*PLUS built-in functions as well.

•Declare section starts with **DECLARE** keyword in which variables, constants, records as cursors can be declared which stores data temporarily. It basically consists definition of PL/SQL identifiers. This part of the code is optional.

•Exception section starts with **EXCEPTION** keyword. This section is optional which contains statements that are executed when a run-time error occurs. Any exceptions can be handled in this section.

Variables & Program data:

PL/SQL identifiers

There are several PL/SQL identifiers such as variables, constants, procedures, cursors, triggers etc.

Variables:

Like several other programming languages, variables in PL/SQL must be declared prior to its use. They should have a valid name and data type as well.

Declaring Program Data:

With few exceptions, you must declare your variables and constants before you use them. These declarations are in the declaration section of your PLSQL program.

Your declarations can include variables, constants, TYPEs (such as collection types or record types), and exceptions.

Declaring a Variable:

When you declare a variable, PL/SQL allocates memory for the variable's value and names the storage location so that the value can be retrieved and changed. The declaration also specifies the datatype of the variable; this datatype is then used to validate values assigned to the variable.

The basic syntax for a declaration is:

name datatype [NOT NULL] [:= | DEFAULT *default_assignment*];

•**Assignment operator (:=)** : It is used to assign a value to a variable.

Example to show how to declare variables in PL/SQL :

```
SQL> SET SERVEROUTPUT ON;

SQL> DECLARE
    var1 INTEGER;
    var2 REAL;
    var3 varchar2(20) ;

BEGIN
    null;
END;
/
```

Output:

```
PL/SQL procedure successfully completed.
```

Explanation:

•**SET SERVEROUTPUT ON**: It is used to display the buffer used by the dbms_ output.

•**var1 INTEGER :** It is the declaration of variable, named ***var1*** which is of integer type. There are many other data types that can be used like float, int, real, smallint, long etc. It also supports variables used in SQL as well like NUMBER(prec, scale), varchar, varchar2 etc.

•**PL/SQL procedure successfully completed.:** It is displayed when the code is compiled and executed successfully.

•**Slash (/) after END;:** The slash (/) tells the SQL*Plus to execute the block.

-------X-------

Procedure to execute pl/sql programs:

- After connection of run sql -> type: set serveroutput on;
- Type ed [filename].sql and open notepad to write a program .
- Save program as [filename].sql in D:\drive.
- Exit from notepad and return to run sql.
- Type @ d:\[filename].sql -> to run the program.
- Type ed d:\[filename].sql -> to correct errors.
- After execution of program
  Type / -> to execute the program again.

❖ -- used for single line comments.

❖ /*[comments]*/ used for multiline comments.

Conditional, Sequential, and loop control statements:

PL/SQL Control Statements:

PL/SQL has three categories of control statements: conditional selection statements, loop statements and sequential control statements.

PL/SQL categories of control statements are:

•**Conditional selection statements**, which run different statements for different data values.

The conditional selection statements are IF and CASE.

•**Loop statements**, which run the same statements with a series of different data values.

The loop statements are the basic LOOP, FOR LOOP, and WHILE LOOP.

The EXIT statement transfers control to the end of a loop. The CONTINUE statement exits the current iteration of a loop and transfers control to the next iteration.

Both EXIT and CONTINUE have an optional WHEN clause, where you can specify a condition.

•**Sequential control statements**, which are not crucial to PL/SQL programming.

The sequential control statements are GOTO, which goes to a specified statement, and NULL, which does nothing.

1)Conditional Selection Statements:

The **conditional selection statements**, IF and CASE, run different statements for different data values.

The IF statement either runs or skips a sequence of one or more statements, depending on a condition. The IF statement has these forms:

•IF THEN

•IF THEN ELSE

•IF THEN ELSIF

a) IF THEN Statement:

The IF THEN statement either runs or skips a sequence of one or more statements, depending on a condition.

Syntax:

```
IF condition THEN
    statements
 END IF;
```

b) IF THEN ELSE Statement: If the value of *condition* is true,the *statements* run; otherwise, the *else_statements* run

syntax:

```
IF condition THEN
        statements
ELSE
       statements
END IF;
```

c) IF THEN ELSIF Statement:The IF THEN ELSIF statement runs the first *statements* for which *condition* is true. Remaining conditions are not evaluated. If no *condition* is true, the *else_statements* run, if they exist; otherwise, the IF THEN ELSIF statement does nothing.

Syntax:

```
IF condition_1 THEN
        statements_1
ELSIF condition_2 THEN
        statements_2
 ELSIF condition_3 THEN
        statements_3
...
ELSE
      statements
END IF;
```

d) Simple CASE Statement:The simple CASE statement runs the first *statements* for which *selector_value* equals *selector*. Remaining conditions are not evaluated. If no *selector_value* equals *selector*, the CASE statement runs *else_statements* if they exist and raises the predefined exception CASE_NOT_FOUND otherwise

syntax:

```
CASE selector
WHEN selector_value_1 THEN statements_1
WHEN selector_value_2 THEN statements_2
...
WHEN selector_value_n THEN statements_n
ELSE
      statements
END CASE;
```

2)LOOP Statements:

Loop statements run the same statements with a series of different values. The loop statements are:

- Basic LOOP
- FOR LOOP
- WHILE LOOP
- Cursor FOR LOOP

The statements that exit a loop are:

- EXIT
- EXIT WHEN

The statements that exit the current iteration of a loop are:

- CONTINUE
- CONTINUE WHEN

EXIT, EXIT WHEN, CONTINUE, and CONTINUE WHEN and can appear anywhere inside a loop,

a) Basic LOOP Statement:

syntax:

[*label*] LOOP
    statements
END LOOP [*label*];

b) FOR LOOP Statement:

The FOR LOOP statement runs one or more statements while the loop index is in a specified range.

Syntax:

[*label*] FOR *index* IN [REVERSE] *lower_bound*..*upper_bound* LOOP
    statements
END LOOP [*label*];

c) WHILE LOOP Statement:

The WHILE LOOP statement runs one or more statements while a condition is true.

syntax:

[*label*] WHILE *condition* LOOP
    statements
END LOOP [*label*];

3)Sequential Control Statements:

Unlike the IF and LOOP statements, the **sequential control statements** GOTO and NULL are not crucial to PL/SQL programming. The GOTO statement, which goes to a specified statement, is seldom needed Occasionally, it simplifies logic enough to warrant its use. The NULL statement, which does nothing, can improve readability by making the meaning and action of conditional statements clear.

Types:

- **GOTO Statement**
- **NULL Statement**

a) GOTO Statement:

The GOTO statement transfers control to a label unconditionally. The label must be unique in its scope and must precede an executable statement or a PL/SQL block. When run, the GOTO statement transfers control to the labeled statement or block.

b) NULL Statement:

The NULL statement only passes control to the next statement. Some languages refer to such an instruction as a no-op (no operation).

-------X-------