Beginning with PL / SQL :

PL/SQL is a combination of SQL along with the procedural features of programming languages. It was developed by Oracle Corporation in the early 90's to enhance the capabilities of SQL. PL/SQL is one of three key programming languages embedded in the Oracle Database, along with SQL itself and Java. This tutorial will give you great understanding on PL/SQL to proceed with Oracle database and other advanced RDBMS concepts.

The PL/SQL programming language was developed by Oracle Corporation in the late 1980s as procedural extension language for SQL and the Oracle relational database. Following are certain notable facts about PL/SQL –

- PL/SQL is a completely portable, high-performance transaction-processing language.
- PL/SQL provides a built-in, interpreted and OS independent programming environment.
- PL/SQL can also directly be called from the command-line **SQL*Plus interface**.
- Direct call can also be made from external programming language calls to database.
- PL/SQL's general syntax is based on that of ADA and Pascal programming language.
- Apart from Oracle, PL/SQL is available in **TimesTen in-memory database** and **IBM DB2**.

Features of PL/SQL

PL/SQL has the following features –

- PL/SQL is tightly integrated with SQL.
- It offers extensive error checking.
- It offers numerous data types.
- It offers a variety of programming structures.
- It supports structured programming through functions and procedures.
- It supports object-oriented programming.
- It supports the development of web applications and server pages.

Advantages of PL/SQL

PL/SQL has the following advantages –

- SQL is the standard database language and PL/SQL is strongly integrated with SQL. PL/SQL supports both static and dynamic SQL. Static SQL supports DML operations and transaction control from PL/SQL block. In Dynamic SQL, SQL allows embedding DDL statements in PL/SQL blocks.
- PL/SQL allows sending an entire block of statements to the database at one time. This reduces network traffic and provides high performance for the applications.
- PL/SQL gives high productivity to programmers as it can query, transform, and update data in a database.
- PL/SQL saves time on design and debugging by strong features, such as exception handling, encapsulation, data hiding, and object-oriented data types.
- Applications written in PL/SQL are fully portable.
- PL/SQL provides high security level.
- PL/SQL provides access to predefined SQL packages.
- PL/SQL provides support for Object-Oriented Programming.
- PL/SQL provides support for developing Web Applications and Server Pages.

The PL/SQL Identifiers

PL/SQL identifiers are constants, variables, exceptions, procedures, cursors, and reserved words. The identifiers consist of a letter optionally followed by more letters, numerals, dollar signs, underscores, and number signs and should not exceed 30 characters.

By default, **identifiers are not case-sensitive**. So you can use **integer** or **INTEGER** to represent a numeric value. You cannot use a reserved keyword as an identifier.

The PL/SQL Delimiters

A delimiter is a symbol with a special meaning. Following is the list of delimiters in PL/SQL –

Delimiter	Description
+, -, *, /	Addition, subtraction/negation, multiplication, division
%	Attribute indicator
'	Character string delimiter
.	Component selector
(,)	Expression or list delimiter
:	Host variable indicator
,	Item separator
"	Quoted identifier delimiter
=	Relational operator
@	Remote access indicator
;	Statement terminator
:=	Assignment operator
=>	Association operator
||	Concatenation operator
**	Exponentiation operator
<<, >>	Label delimiter (begin and end)
/*, */	Multi-line comment delimiter (begin and end)
--	Single-line comment indicator
..	Range operator
<, >, <=, >=	Relational operators
<>, '=, ~=, ^=	Different versions of NOT EQUAL

The PL/SQL Comments

Program comments are explanatory statements that can be included in the PL/SQL code that you write and helps anyone reading its source code. All programming languages allow some form of comments.

The PL/SQL supports single-line and multi-line comments. All characters available inside any comment are ignored by the PL/SQL compiler. The PL/SQL single-line comments start with the delimiter -- (double hyphen) and multi-line comments are enclosed by /* and */.

```
DECLARE
  -- variable declaration
  message  varchar2(20):= 'Hello, World!';
BEGIN
  /*
  *  PL/SQL executable statement(s)
  */
  dbms_output.put_line(message);
```

```
END;
/
```

When the above code is executed at the SQL prompt, it produces the following result −

```
Hello World

PL/SQL procedure successfully completed.
```

PL/SQL Program Units

A PL/SQL unit is any one of the following −

- PL/SQL block
- Function
- Package
- Package body
- Procedure
- Trigger
- Type
- Type body

PL/SQL - Operators

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulation. PL/SQL language is rich in built-in operators and provides the following types of operators −

- Arithmetic operators
- Relational operators
- Comparison operators
- Logical operators
- String operators

1.Arithmetic Operators

Following table shows all the arithmetic operators supported by PL/SQL. Let us assume **variable A** holds 10 and **variable B** holds 5, then −

Show Examples

Operator	Description	Example
+	Adds two operands	A + B will give 15
-	Subtracts second operand from the first	A - B will give 5
*	Multiplies both operands	A * B will give 50
/	Divides numerator by de-numerator	A / B will give 2
**	Exponentiation operator, raises one operand to the power of other	A ** B will give 100000

Relational Operators

Relational operators compare two expressions or values and return a Boolean result. Following table shows all the relational operators supported by PL/SQL. Let us assume **variable A** holds 10 and **variable B** holds 20, then –

Show Examples

Operator	Description	Example
=	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(A = B) is not true.
!= <> ~=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true

2. Comparison Operators

Comparison operators are used for comparing one expression to another. The result is always either **TRUE, FALSE** or **NULL**.

Show Examples

Operator	Description	Example
LIKE	The LIKE operator compares a character, string, or CLOB value to a pattern and returns TRUE if the value matches the pattern and FALSE if it does not.	If 'Zara Ali' like 'Z% A_i' returns a Boolean true, whereas, 'Nuha Ali' like 'Z% A_i' returns a Boolean false.
BETWEEN	The BETWEEN operator tests whether a value lies in a specified range. x BETWEEN a AND b means that x >= a and x <= b.	If x = 10 then, x between 5 and 20 returns true, x between 5 and 10 returns true, but x between 11 and 20 returns false.
IN	The IN operator tests set membership. x IN (set) means that x is equal to any member of set.	If x = 'm' then, x in ('a', 'b', 'c') returns Boolean false but x in ('m', 'n', 'o') returns Boolean true.
IS NULL	The IS NULL operator returns the BOOLEAN value TRUE if its operand is NULL or FALSE if it is not NULL. Comparisons involving NULL values always yield NULL.	If x = 'm', then 'x is null' returns Boolean false.

3.Logical Operators

Following table shows the Logical operators supported by PL/SQL. All these operators work on Boolean operands and produce Boolean results. Let us assume **variable A** holds true and **variable B** holds false, then −

Show Examples

Operator	Description	Examples
and	Called the logical AND operator. If both the operands are true then condition becomes true.	(A and B) is false.
or	Called the logical OR Operator. If any of the two operands is true then condition becomes true.	(A or B) is true.
not	Called the logical NOT Operator. Used to reverse the logical state of its operand. If a condition is true then Logical NOT operator will make it false.	not (A and B) is true.

4. PL/SQL Operator Precedence

Operator precedence determines the grouping of terms in an expression. This affects how an expression is evaluated. Certain operators have higher precedence than others; for example, the multiplication operator has higher precedence than the addition operator.

For example, **x = 7 + 3 * 2**; here, **x** is assigned **13**, not 20 because operator * has higher precedence than +, so it first gets multiplied with **3*2** and then adds into **7**.

Here, operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom. Within an expression, higher precedence operators will be evaluated first.

The precedence of operators goes as follows: =, <, >, <=, >=, <>, !=, ~=, ^=, IS NULL, LIKE, BETWEEN, IN.

Show Examples

Operator	Operation
**	exponentiation
+, -	identity, negation
*, /	multiplication, division
+, -, ||	addition, subtraction, concatenation
comparison	
NOT	logical negation
AND	conjunction
OR	inclusion

Expressions:

Expressions are constructed by using operands and operators.

An operand is a variable, constant, literal, or function call that contributes a value to an expression.

An operator like + or -indicates what the program should do with the operands.

Sequences:
https://www.techonthenet.com/oracle/sequences.php

Control Structures:

The IF-THEN statement of PL/SQL language has the same structure as the others equivalent procedural languages. The IF-THEN statement allows selective execution of actions based on the fulfillment of certain conditions.

IF-THEN Statement

If you want to execute a certain section of code only if a condition is true, you can use an IF-THEN statement.

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

IF-THEN-ELSIF Statement

You can also use an IF-THEN-ELSIF statement to check multiple conditions.

```
IF condition_1 THEN
   -- statement_1
ELSIF condition_2 THEN
   -- statement_2
END IF;
```

IF-THEN-ELSE Statement

If you want to execute a certain section of code only if a condition is NOT true, you can use an IF-THEN-ELSE statement.

```
IF condition_1 THEN
   -- statement_1
ELSIF condition_2 THEN
   -- statement_2
```

```
ELSE
   -- statement_3
END IF;
```

CASE Statement

You can use the CASE statement to select one of several alternative sections of code to execute.

```
CASE expression
   WHEN condition_1 THEN statement_1;
   WHEN condition_2 THEN statement_2;
   WHEN condition_3 THEN statement_3;
   ELSE statement_4;
END CASE;
```

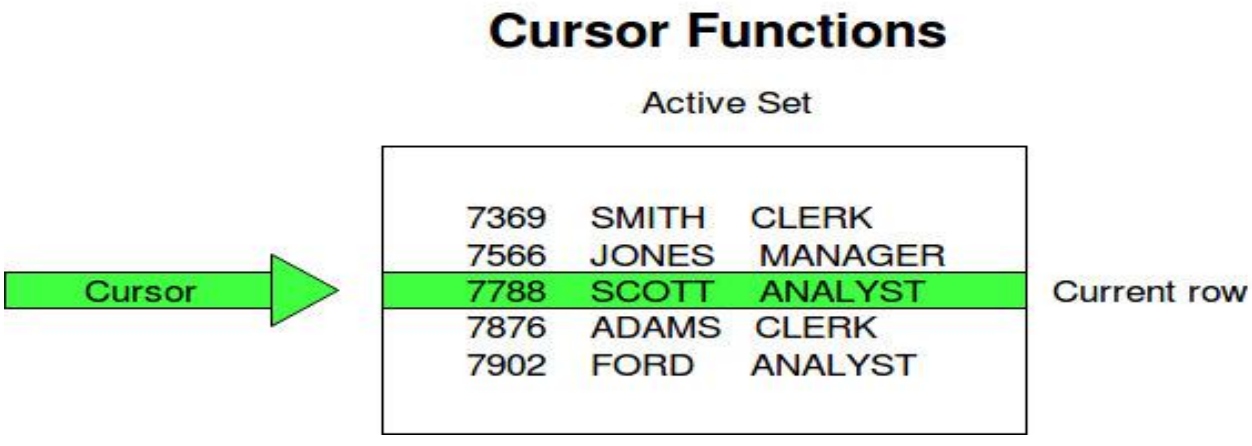
Cursors and Transaction 👍

Cursor in SQL

To execute SQL statements, a work area is used by the Oracle engine for its internal processing and storing the information. This work area is private to SQL's operations. The 'Cursor' is the PL/SQL construct that allows the user to name the work area and access the stored information in it.

Use of Cursor

The major function of a cursor is to retrieve data, one row at a time, from a result set, unlike the SQL commands which operate on all the rows in the result set at one time. Cursors are used when the user needs to update records in a singleton fashion or in a row by row manner, in a database table.
The Data that is stored in the Cursor is called the *Active Data Set*. Oracle DBMS has another predefined area in the main memory Set, within which the cursors are opened. Hence the size of the cursor is limited by the size of this pre-defined area.



Cursor Actions

- Declare Cursor: A cursor is declared by defining the SQL statement that returns a result set.
- Open: A Cursor is opened and populated by executing the SQL statement defined by the cursor.
- Fetch: When the cursor is opened, rows can be fetched from the cursor one by one or in a block to perform data manipulation.
- Close: After data manipulation, close the cursor explicitly.
- Deallocate: Finally, delete the cursor definition and release all the system resources associated with the cursor.

Types of Cursors
Cursors are classified depending on the circumstances in which they are opened.

- **Implicit Cursor:** If the Oracle engine opened a cursor for its internal processing it is known as an Implicit Cursor. It is created "automatically" for the user by Oracle when a query is executed and is simpler to code.
- **Explicit Cursor:** A Cursor can also be opened for processing data through a PL/SQL block, on demand. Such a user-defined cursor is known as an Explicit Cursor.

Collections and composite data types:

The PL/SQL variables, constants and parameters must have a valid data type, which specifies a storage format, constraints, and a valid range of values.

S.No	Category & Description
1	**Scalar** Single values with no internal components, such as a **NUMBER, DATE,** or **BOOLEAN**.
2	**Large Object (LOB)** Pointers to large objects that are stored separately from other data items, such as text, graphic images, video clips, and sound waveforms.
3	**Composite** Data items that have internal components that can be accessed individually. For example, collections and records.
4	**Reference** Pointers to other data items.

PL/SQL Scalar Data Types and Subtypes

PL/SQL Scalar Data Types and Subtypes come under the following categories –

S.No	Date Type & Description
1	**Numeric** Numeric values on which arithmetic operations are performed.
2	**Character** Alphanumeric values that represent single characters or strings of characters.
3	**Boolean** Logical values on which logical operations are performed.
4	**Datetime** Dates and times.

PL/SQL provides subtypes of data types. For example, the data type NUMBER has a subtype called INTEGER. You can use the subtypes in your PL/SQL program to make the data types compatible with data types in other programs while embedding the PL/SQL code in another program, such as a Java program.

PL/SQL Numeric Data Types and Subtypes

Following table lists out the PL/SQL predefined numeric data types and their subtypes –

S.No	Data Type & Description
1	**PLS_INTEGER** Signed integer in range -2,147,483,648 through 2,147,483,647, represented in 32 bits
2	**BINARY_INTEGER** Signed integer in range -2,147,483,648 through 2,147,483,647, represented in 32 bits
3	**BINARY_FLOAT** Single-precision IEEE 754-format floating-point number
4	**BINARY_DOUBLE** Double-precision IEEE 754-format floating-point number
5	**NUMBER(prec, scale)** Fixed-point or floating-point number with absolute value in range 1E-130 to (but not including) 1.0E126. A NUMBER variable can also represent 0

6	**DEC(prec, scale)** ANSI specific fixed-point type with maximum precision of 38 decimal digits
7	**DECIMAL(prec, scale)** IBM specific fixed-point type with maximum precision of 38 decimal digits
8	**NUMERIC(pre, scale)** Floating type with maximum precision of 38 decimal digits
9	**DOUBLE PRECISION** ANSI specific floating-point type with maximum precision of 126 binary digits (approximately 38 decimal digits)
10	**FLOAT** ANSI and IBM specific floating-point type with maximum precision of 126 binary digits (approximately 38 decimal digits)
11	**INT** ANSI specific integer type with maximum precision of 38 decimal digits
12	**INTEGER** ANSI and IBM specific integer type with maximum precision of 38 decimal digits
13	**SMALLINT** ANSI and IBM specific integer type with maximum precision of 38 decimal digits
14	**REAL** Floating-point type with maximum precision of 63 binary digits (approximately 18 decimal digits)

https://www.tutorialspoint.com/plsql/plsql_data_types.htm

Procedures and Functions:

SQL (Structured Query Language) is a computer language which is used to interact with an **RDBMS** (Relational Database Management System). It is basically a method of managing, organizing, and retrieving data from a relation database. In SQL, two important concepts are used namely, **function** and **procedure**.

A function calculates the results of a program based on the inputs provided, whereas a procedure is used to perform some tasks in a specific order. There

are many other differences between functions and procedures, which we will discuss in this article.

What is Function?

A **function**, in the context of computer programming languages, is a set of instructions which takes some input and performs certain tasks. In SQL, a function returns a value. In other words, a function is a tool in SQL that is used to calculate anything to produce an output for the provided inputs. In SQL queries, when a function is called, it returns the resulting value. It also controls to the calling function. However, in a function, we cannot use some DML statements like Insert, Delete, Update, etc.

Also, a function can be called through a procedure. Based on definition, there are two types of functions namely, **predefined function** and **userdefined function**. Another important point about functions is that they may or may not return a value, i.e. a function can return a null valued as well.

What is a Procedure?

A **procedure** is a set of instructions which takes input and performs a certain task. In SQL, procedures do not return a value. In Java, procedures and functions are same and also called **subroutines**.

In SQL, a procedure is basically a precompiled statement which is stored inside the database. Therefore, a procedure is sometimes also called a **stored procedure**. A procedure always has a name, list of parameters, and compiled SQL statements. In SQL, a procedure does not return any value.

Difference between Function and Procedure

Following are the important differences between SQL Function and SQL Procedure−

Key	Function	Procedure
Definition	A function is used to calculate result using given inputs.	A procedure is used to perform certain task in order.
Call	A function can be called by a procedure.	A procedure cannot be called by a function.
DML	DML statements cannot be executed within a function.	DML statements can be executed within a procedure.
SQL, Query	A function can be called within a query.	A procedure cannot be called within a query.
SQL, Call	Whenever a function is called, it is first compiled before being called.	A procedure is compiled once and can be called multiple times without being compiled.
SQL, Return	A function returns a value and control to calling function or code.	A procedure returns the control but not any value to calling function or code.
try-catch	A function has no support for try-catch	A procedure has support for try-catch blocks.
SELECT	A select statement can have a function call.	A select statement can't have a procedure call.
Explicit Transaction Handling	A function cannot have explicit transaction handling.	A procedure can use explicit transaction handling.

Exceptions Handling:
https://www.geeksforgeeks.org/exception-handling-plsql/

Packages and Triggers:

https://www.tutorialspoint.com/plsql/plsql_packages.htm

https://www.tutorialspoint.com/plsql/plsql_triggers.htm