

Introduction to PL/SQL

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

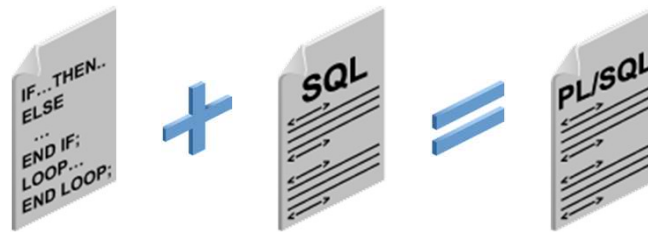
- Explain the need for PL/SQL
- Explain the benefits of PL/SQL
- Identify the different types of PL/SQL blocks
- Output messages in PL/SQL

This lesson introduces PL/SQL and the PL/SQL programming constructs. You also learn about the benefits of PL/SQL.

About PL/SQL

PL/SQL:

- Stands for “Procedural Language extension to SQL”
- Is Oracle Corporation’s standard data access language for relational databases
- Seamlessly integrates procedural constructs with SQL



Structured Query Language (SQL) is the primary language used to access and modify data in relational databases. There are only a few SQL commands, so you can easily learn and use them.

Consider an example:

```
SELECT first_name, department_id, salary FROM employees;
```

The preceding SQL statement is simple and straightforward. However, if you want to alter any data that is retrieved in a conditional manner, you soon encounter the limitations of SQL.

Consider a slightly modified problem statement: For every employee retrieved, check the department ID and salary. Depending on the department’s performance and also the employee’s salary, you may want to provide varying bonuses to the employees.

Looking at the problem, you know that you have to execute the preceding SQL statement, collect the data, and apply logic to the data.

- One solution is to write a SQL statement for each department to give bonuses to the employees in that department. Remember that you also have to check the salary component before deciding the bonus amount. This makes it a little complicated.
- A more effective solution might include conditional statements. PL/SQL is designed to meet such requirements. It provides a programming extension to the already-existing SQL.

About PL/SQL

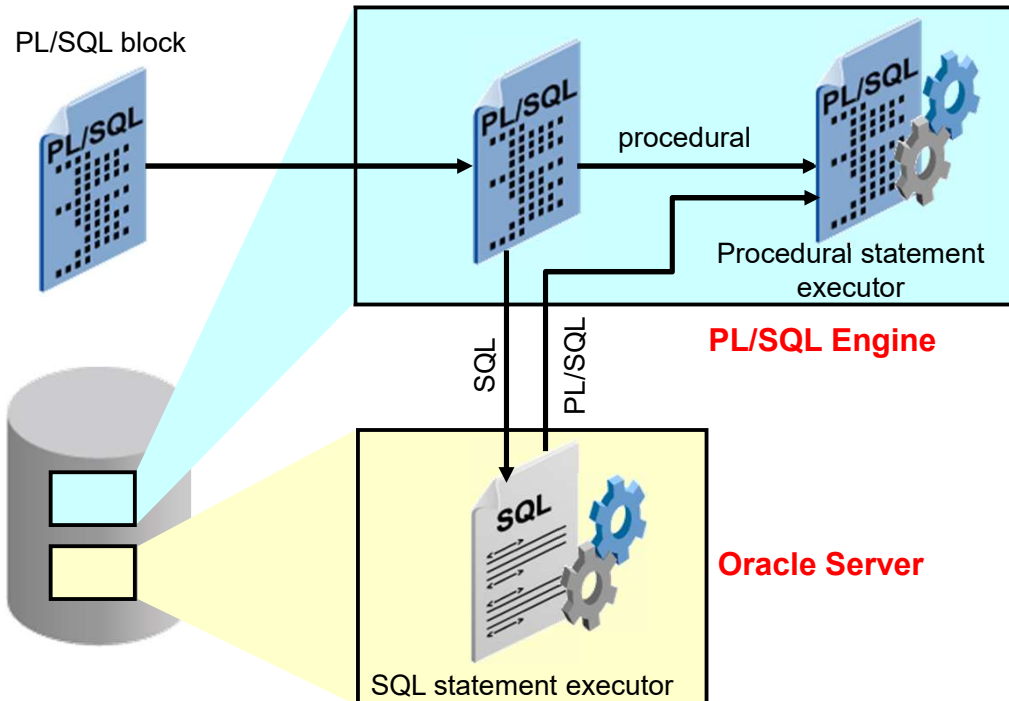
PL/SQL:

- Provides a block structure for executable units of code. Maintenance of code is made easier with such a well-defined structure.
- Provides procedural constructs such as:
 - Variables, constants, and data types
 - Control structures such as conditional statements and loops
 - Reusable program units that are written once and executed many times

PL/SQL defines a block structure for writing code. Maintaining and debugging code is made easier with such a structure because you can easily understand the flow and execution of the program unit.

PL/SQL offers modern software engineering features such as data encapsulation, exception handling, information hiding, and object orientation. It brings state-of-the-art programming to the Oracle Server and toolset. PL/SQL provides all the procedural constructs that are available in any third-generation language (3GL).

PL/SQL Run-Time Architecture



The diagram in the slide shows a PL/SQL block being executed by the PL/SQL engine. The PL/SQL engine resides in:

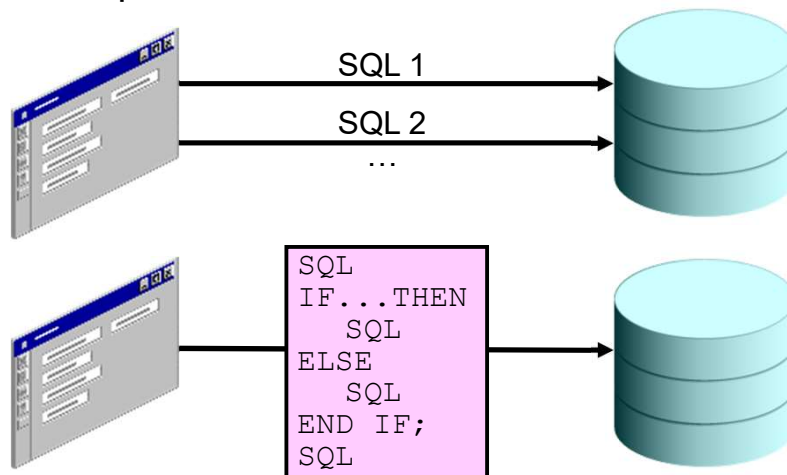
- The Oracle database for executing stored subprograms
- The Oracle Forms client when you run client/server applications, or in the Oracle Application Server when you use Oracle Forms Services to run Forms on the web

Irrespective of the PL/SQL run-time environment, the basic architecture remains the same. Therefore, all PL/SQL statements are processed in the Procedural Statement Executor, and all SQL statements must be sent to the SQL Statement Executor for processing by the Oracle Server processes. The SQL environment may also invoke the PL/SQL environment. For example, the PL/SQL environment is invoked when a PL/SQL function is used in a `SELECT` statement.

The PL/SQL engine is a virtual machine that resides in memory and processes the PL/SQL m-code instructions. When the PL/SQL engine encounters a SQL statement, a context switch is made to pass the SQL statement to the Oracle Server processes. The PL/SQL engine waits for the SQL statement to complete and for the results to be returned before it continues to process subsequent statements in the PL/SQL block. The Oracle Forms PL/SQL engine runs in the client for the client/server implementation, and in the application server for the Forms Services implementation. In either case, SQL statements are typically sent over a network to an Oracle Server for processing.

Benefits of PL/SQL

- Integration of procedural constructs with SQL
- Improved performance



- **Integration of procedural constructs with SQL:** The most important advantage of PL/SQL is the integration of procedural constructs with SQL. SQL is a nonprocedural language. When you issue a SQL command, your command tells the database server *what* to do. However, you cannot specify *how* to do it. PL/SQL integrates control statements and conditional statements with SQL, giving you better control of your SQL statements and their execution. Earlier in this lesson, you saw an example of the need for such integration.
- **Improved performance:** Without PL/SQL, you would not be able to logically combine SQL statements as one unit. If you have designed an application that contains forms, you may have many different forms with fields in each form. When a form submits data, you may have to execute a number of SQL statements. SQL statements are sent to the database one at a time. This results in many network trips and one call to the database for each SQL statement, thereby increasing network traffic and reducing performance (especially in a client/server model).

With PL/SQL, you can combine all these SQL statements into a single program unit. The application can send the entire block to the database instead of sending the SQL statements one at a time. This significantly reduces the number of database calls. As the slide illustrates, if the application is SQL intensive, you can use PL/SQL blocks to group SQL statements before sending them to the Oracle database server for execution.

Benefits of PL/SQL

- **Modularized program development**
 - Blocks can be in a sequence or they can be nested in other blocks.
- **Integration with Oracle tools**
 - The PL/SQL engine is integrated in Oracle tools such as Oracle Forms and Oracle Reports
- **Portability**
 - PL/SQL programs can run anywhere an Oracle Server runs, irrespective of the operating system and platform.
- **Exception handling**
 - . You can define separate blocks for dealing with exceptions.

ORACLE

- **Modularized program development:** The basic unit in all PL/SQL programs is the block.
- Modularized program development has the following advantages:
 - You can group logically related statements within blocks.
 - You can nest blocks inside larger blocks to build powerful programs.
 - You can break your application into smaller modules. If you are designing a complex application, PL/SQL allows you to break down the application into smaller, manageable, and logically related modules.
 - You can easily maintain and debug code.

In PL/SQL, modularization is implemented using procedures, functions, and packages, which are discussed in the lesson titled “Introducing Stored Procedures and Functions.”

- **Integration with tools:** The PL/SQL engine is integrated in Oracle tools such as Oracle Forms and Oracle Reports. When you use these tools, the locally available PL/SQL engine processes the procedural statements; only the SQL statements are passed to the database.

- **Portability:** PL/SQL programs can run anywhere an Oracle Server runs, irrespective of the operating system and platform. You do not need to customize them to each new environment. You can write portable program packages and create libraries that can be reused in different environments.
- **Exception handling:** PL/SQL enables you to handle exceptions efficiently. You can define separate blocks for dealing with exceptions. You learn more about exception handling in the lesson titled “Handling Exceptions.”

PL/SQL shares the same data type system as SQL (with some extensions) and uses the same expression syntax.

PL/SQL Block Structure

- **DECLARE (optional)**
 - Variables, cursors, user-defined exceptions
- **BEGIN (mandatory)**
 - SQL statements
 - PL/SQL statements
- **EXCEPTION (optional)**
 - Actions to perform when exceptions occur
- **END; (mandatory)**



The slide shows a basic PL/SQL block. A PL/SQL block consists of four sections:

- **Declarative (optional):** The declarative section begins with the keyword `DECLARE` and ends when the executable section starts.
- **Begin (required):** The executable section begins with the keyword `BEGIN`. This section needs to have at least one statement. However, the executable section of a PL/SQL block can include any number of PL/SQL blocks.
- **Exception handling (optional):** The exception section is nested within the executable section. This section begins with the keyword `EXCEPTION`.
- **End (required):** All PL/SQL blocks must conclude with an `END` statement. Observe that `END` is terminated with a semicolon.
- In a PL/SQL block, the keywords `DECLARE`, `BEGIN`, and `EXCEPTION` are not terminated by a semicolon. However, the keyword `END`, all SQL statements, and PL/SQL statements must be terminated with a semicolon.

Block Types

Anonymous

```
[DECLARE]

BEGIN
    --statements

[EXCEPTION]

END;
```

Procedure

```
PROCEDURE name
IS
BEGIN
    --statements

[EXCEPTION]

END;
```

Function

```
FUNCTION name
RETURN datatype
IS
BEGIN
    --statements
    RETURN value;
[EXCEPTION]

END;
```

A PL/SQL program comprises one or more blocks. These blocks can be entirely separate or nested within another block.

There are three types of blocks that make up a PL/SQL program:

- Anonymous blocks
- Procedures
- Functions

Anonymous blocks: Anonymous blocks are unnamed blocks. They are declared inline at the point in an application where they are to be executed and are compiled each time the application is executed. These blocks are not stored in the database. They are passed to the PL/SQL engine for execution at run time. Triggers in Oracle Developer components consist of such blocks.

If you want to execute the same block again, you have to rewrite the block. You cannot invoke or call the block that you wrote earlier because blocks are anonymous and do not exist after they are executed.

Procedures: Procedures are named objects that contain SQL and/or PL/SQL statements.

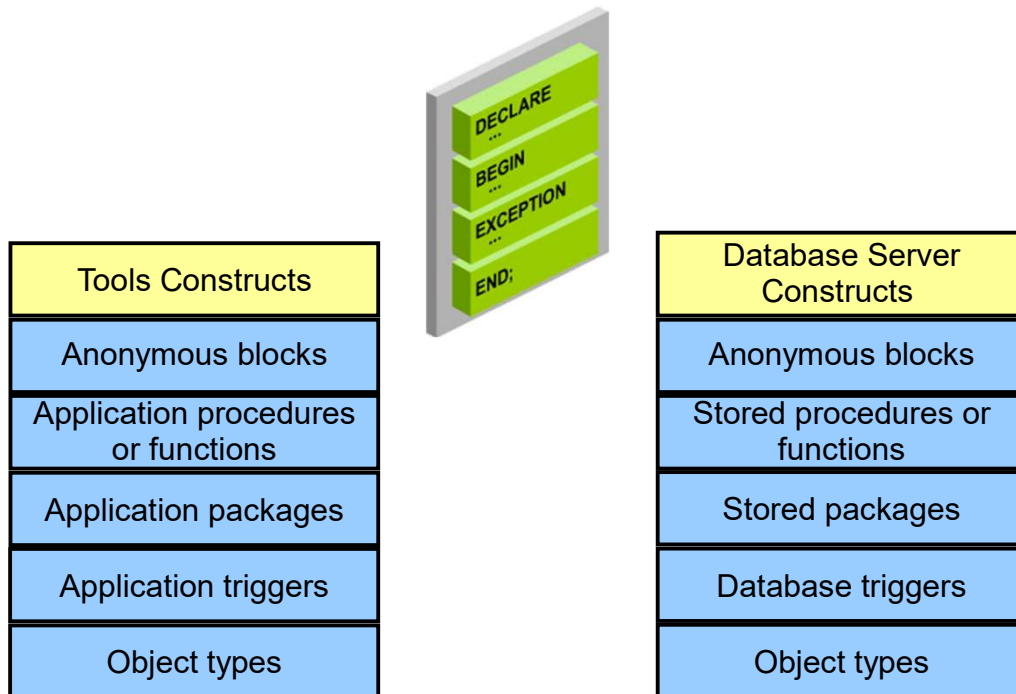
Functions: Functions are named objects that contain SQL and/or PL/SQL statements. Unlike a procedure, a function returns a value of a specified data type.

Subprograms

Subprograms are complementary to anonymous blocks. They are named PL/SQL blocks that are stored in the database. Because they are named and stored, you can invoke them whenever you want (depending on your application). You can declare them either as procedures or as functions. You typically use a procedure to perform an action and a function to compute and return a value.

Subprograms can be stored at the server or application level. Using Oracle Developer components (Forms, Reports), you can declare procedures and functions as part of the application (a form or report) and call them from other procedures, functions, and triggers within the same application, whenever necessary.

Program Constructs



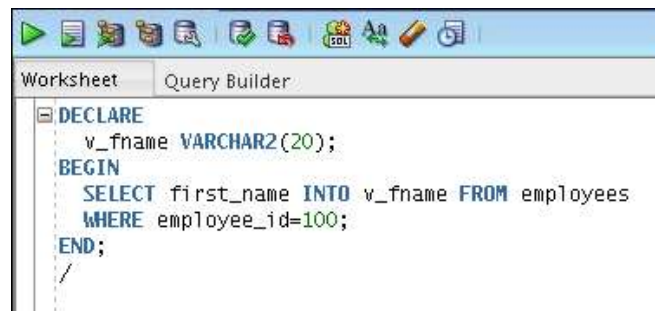
ORACLE

The following table outlines a variety of PL/SQL program constructs that use the basic PL/SQL block. The program constructs are available based on the environment in which they are executed.

Program Construct	Description	Availability
Anonymous blocks	Unnamed PL/SQL blocks that are embedded within an application or are issued interactively	All PL/SQL environments
Application procedures or functions	Named PL/SQL blocks that are stored in an Oracle Forms Developer application or a shared library; can accept parameters and can be invoked repeatedly by name	Oracle Developer tools components (for example, Oracle Forms Developer, Oracle Reports)
Stored procedures or functions	Named PL/SQL blocks that are stored in the Oracle server; can accept parameters and can be invoked repeatedly by name	Oracle server or Oracle Developer tools
Packages (application or stored)	Named PL/SQL modules that group related procedures, functions, and identifiers	Oracle server and Oracle Developer tools components (for example, Oracle Forms Developer)

Examining an Anonymous Block

An anonymous block in the SQL Developer workspace:

A screenshot of the SQL Developer workspace. The top toolbar contains various icons for file operations, execution, and development. Below the toolbar, there are two tabs: 'Worksheet' and 'Query Builder'. The 'Worksheet' tab is active, displaying an anonymous SQL block. The code is as follows:

```
DECLARE
  v_fname VARCHAR2(20);
BEGIN
  SELECT first_name INTO v_fname FROM employees
  WHERE employee_id=100;
END;
```

To create an anonymous block by using SQL Developer, enter the block in the workspace (as shown in the slide).

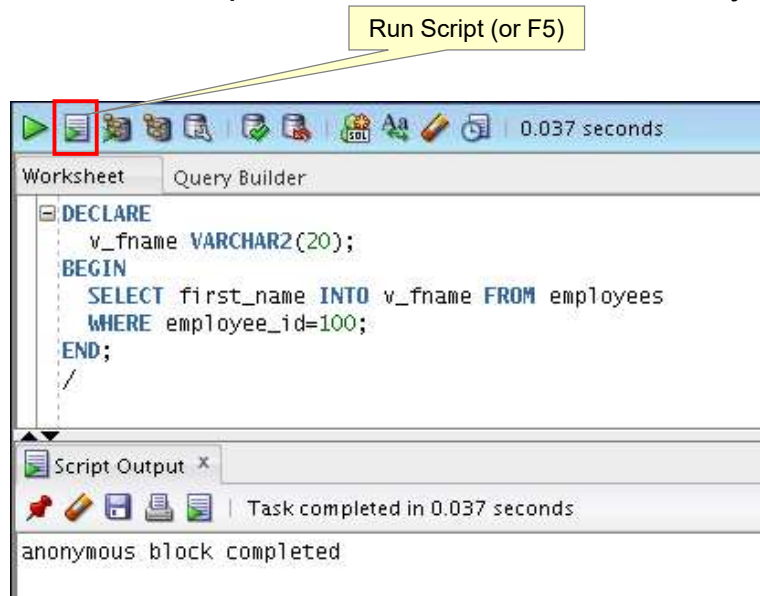
Example

The example block has the declarative section and the executable section. You need not pay attention to the syntax of statements in the block; you learn the syntax later in the course.

The anonymous block gets the `first_name` of the employee whose `employee_id` is 100, and stores it in a variable called `v_fname`.

Executing an Anonymous Block

Click the Run Script button to execute the anonymous block:



To execute an anonymous block, click the Run Script button (or press F5).

Note: The message “anonymous block completed” is displayed in the Script Output window after the block is executed.

Enabling Output of a PL/SQL Block

1. To enable output in SQL Developer, execute the following command before running the PL/SQL block:

```
SET SERVEROUTPUT ON
```

2. Use a predefined Oracle package and its procedure in the anonymous block:

– DBMS_OUTPUT.PUT_LINE

```
DBMS_OUTPUT.PUT_LINE(' The First Name of the  
Employee is ' || v_fname);  
...
```

In the example shown in the previous slide, a value is stored in the `v_fname` variable. However, the value has not been printed.

PL/SQL does not have built-in input or output functionality. Therefore, you need to use predefined Oracle packages for input and output. To generate output, you must perform the following:

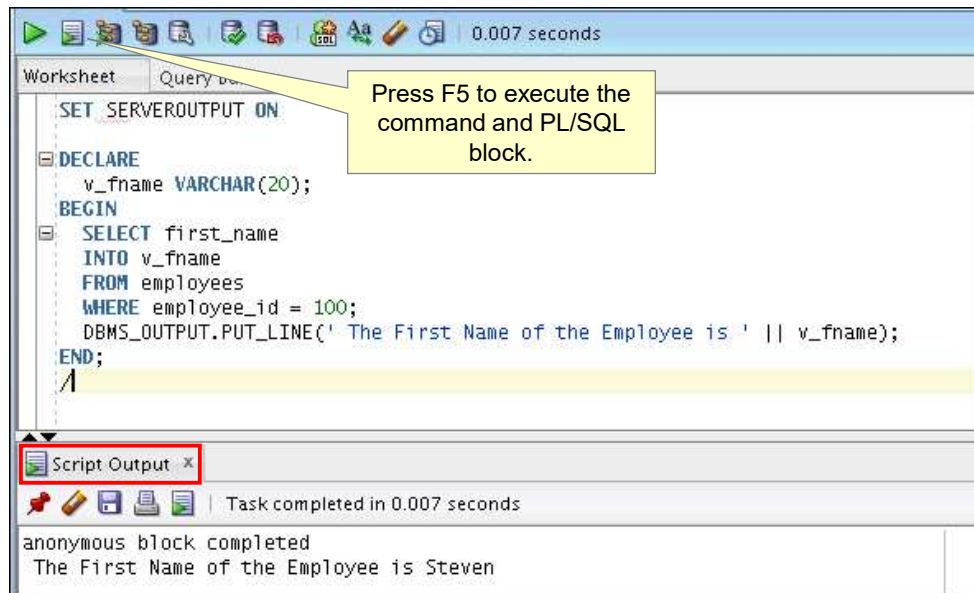
1. Execute the following command:

```
SET SERVEROUTPUT ON
```

Note: To enable output in SQL*Plus, you must explicitly issue the `SET SERVEROUTPUT ON` command.

2. In the PL/SQL block, use the `PUT_LINE` procedure of the `DBMS_OUTPUT` package to display the output. Pass the value that has to be printed as an argument to this procedure (as shown in the slide). The procedure then outputs the argument.

Viewing the Output of a PL/SQL Block



Press F5 (or click the Run Script icon) to view the output for the PL/SQL block. This action:

1. Executes the `SET SERVEROUTPUT ON` command
2. Runs the anonymous PL/SQL block

The output appears on the Script Output tab.

Quiz

A PL/SQL block *must* consist of the following three sections:

- A Declarative section, which begins with the keyword `DECLARE` and ends when the executable section starts
 - An Executable section, which begins with the keyword `BEGIN` and ends with `END`
 - An Exception handling section, which begins with the keyword `EXCEPTION` and is nested within the executable section
- a. True
- b. False

Answer: b

A PL/SQL block consists of three sections:

- **Declarative (optional):** The optional declarative section begins with the keyword `DECLARE` and ends when the executable section starts.
- **Executable (required):** The required executable section begins with the keyword `BEGIN` and ends with `END`. This section essentially needs to have at least one statement. Observe that `END` is terminated with a semicolon. The executable section of a PL/SQL block can, in turn, include any number of PL/SQL blocks.
- **Exception handling (optional):** The optional exception section is nested within the executable section. This section begins with the keyword `EXCEPTION`.

Summary

In this lesson, you should have learned how to:

- Integrate SQL statements with PL/SQL program constructs
- Describe the benefits of PL/SQL
- Differentiate between PL/SQL block types
- Output messages in PL/SQL

PL/SQL is a language that has programming features that serve as extensions to SQL. SQL, which is a nonprocedural language, is made procedural with PL/SQL programming constructs. PL/SQL applications can run on any platform or operating system on which an Oracle Server runs. In this lesson, you learned how to build basic PL/SQL blocks.