|  |
| --- |
| EPAM Systems, RD Dep. |
| MTN.BI.08 Core PL/SQL |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| REVISION HISTORY | | | | | |
| Ver. | Description of Change | Author | Date | Approved | |
| Name | Effective Date |
| 1.0 | Initial status | [Elias Nema](mailto:Elias_Nema@epam.com) | 20-JAN-2014 |  |  |
|  |  |  |  |  |  |

Contents

[1. PL/SQL 3](#_Toc383390304)

[2. Introduction to Server-Side Programming 3](#_Toc383390305)

[2.1. Overview of PL/SQL 4](#_Toc383390306)

[2.1.1. PL/SQL Subprograms 4](#_Toc383390307)

[2.1.2. PL/SQL Packages 4](#_Toc383390308)

[2.2. How PL/SQL Runs 4](#_Toc383390309)

[3. What is Dynamic SQL? 7](#_Toc383390310)

[3.1. Advantages and Disadvantages of Dynamic SQL 7](#_Toc383390311)

[3.2. When to Use Dynamic SQL 7](#_Toc383390312)

[3.3. PL/SQL examples 7](#_Toc383390313)

[4. Source Books and Articles 7](#_Toc383390314)

# PL/SQL

PL/SQL is a procedural extension to Oracle SQL. PL/SQL is integrated with Oracle Database, enabling you to use all of the Oracle Database SQL statements, functions, and data types. You can use PL/SQL to control the flow of a SQL program, use variables, and write error-handling procedures.

A primary benefit of PL/SQL is the ability to store application logic in the database itself. A PL/SQL procedure or function is a schema object that consists of a set of SQL statements and other PL/SQL constructs, grouped together, stored in the database, and run as a unit to solve a specific problem or to perform a set of related tasks. The principal benefit of server-side programming is that built-in functionality can be deployed anywhere.

# Introduction to Server-Side Programming

In a nonprocedural language such as SQL, the set of data to be operated on is specified, but not the operations to be performed or the manner in which they are to be carried out. In a procedural language program, most statement execution depends on previous or subsequent statements and on control structures, such as loops or conditional branches that are not available in SQL.

For an illustration of the difference between procedural and nonprocedural languages, suppose that the following SQL statement queries the employees table:

SELECT employee\_id, department\_id, last\_name, salary FROM employees;

The preceding statement requests data, but does not apply logic to the data. However, suppose you want an application to determine whether each employee in the data set deserves a raise based on salary and department performance. A necessary condition of a raise is that the employee did not receive more than three raises in the last five years. If a raise is called for, then the application must adjust the salary and email the manager; otherwise, the application must update a report.

The problem is how procedural database applications requiring conditional logic and program flow control can use SQL. The basic development approaches are as follows:

* *Use client-side programming to embed SQL statements in applications written in procedural languages such as C, C++, or Java*. You can place SQL statements in source code and submit it to a precompiler or Java translator before compilation. Alternatively, you can eliminate the precompilation step and use an API such as Java Database Connectivity (JDBC) or Oracle Call Interface (OCI) to enable the application to interact with the database.
* *Use server-side programming to develop data logic that resides in the database*. An application can explicitly invoke stored subprograms (procedures and functions), written in PL/SQL (pronounced P L sequel) or Java. You can also create a trigger, which is named program unit that is stored in the database and invoked in response to a specified event.

The principal benefit of server-side programming is that functionality built into the database can be deployed anywhere. The database and not the application determines the best way to perform tasks on a given operating system. Also, subprograms increase scalability by centralizing application processing on the server, enabling clients to reuse code. Because subprogram calls are quick and efficient, a single call can start a compute-intensive stored subprogram, reducing network traffic.

You can use the following languages to store data logic in Oracle Database:

**PL/SQL.** PL/SQL is the Oracle Database procedural extension to SQL. PL/SQL is integrated with the database, supporting all Oracle SQL statements, functions, and data types. Applications written in database APIs can invoke PL/SQL stored subprograms and send PL/SQL code blocks to the database for execution.

**Java.** Oracle Database also provides support for developing, storing, and deploying Java applications. Java stored subprograms run in the database and are independent of programs that run in the middle tier. Java stored subprograms run in the database and are independent of programs that run in the middle tier. Java stored subprograms interface with SQL using a similar execution model to PL/SQL.

## Overview of PL/SQL

PL/SQL provides a server-side, stored procedural language that is easy-to-use, seamless with SQL, robust, portable, and secure. You can access and manipulate database data using procedural schema objects called PL/SQL program units.

PL/SQL program units generally are categorized as follows:

* A PL/SQL subprogram is a PL/SQL block that is stored in the database and can be called by name from an application. When you create a subprogram, the database parses the subprogram and stores its parsed representation in the database. You can declare a subprogram as a procedure or a function.
* A PL/SQL anonymous block is a PL/SQL block that appears in your application and is not named or stored in the database. In many applications, PL/SQL blocks can appear wherever SQL statements can appear.

The PL/SQL compiler and interpreter are embedded in Oracle SQL Developer, giving developers a consistent and leveraged development model on both client and server.

### PL/SQL Subprograms

A PL/SQL subprogram is a named PL/SQL block that permits the caller to supply parameters that can be input only, output only, or input and output values. A subprogram solves a specific problem or performs related tasks and serves as a building block for modular, maintainable database applications.

A subprogram is either a PL/SQL procedure or a PL/SQL function. Procedures and functions are identical except that functions always return a single value to the caller, whereas procedures do not. The term PL/SQL procedure in this chapter refers to either a procedure or a function.

A subprogram created at the schema level with the CREATE PROCEDURE or CREATE FUNCTION statement is a standalone stored subprogram. Subprograms defined in a package are called package subprograms and are considered a part of the package. The database stores subprograms in the data dictionary as schema objects. A subprogram has a specification, which includes descriptions of any parameters, and a body.

### PL/SQL Packages

A PL/SQL package is a group of related subprograms, along with the cursors and variables they use, stored together in the database for continued use as a unit. Packaged subprograms can be called explicitly by applications or users.

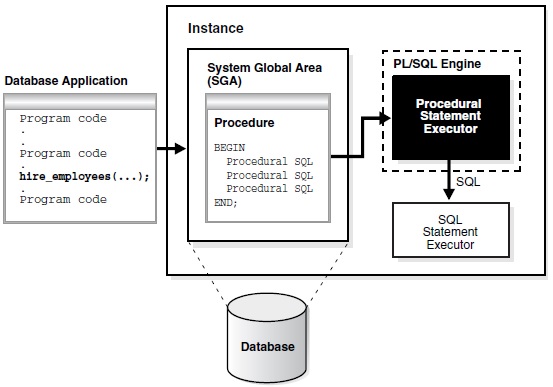
Oracle Database includes many supplied packages that extend database functionality and provide PL/SQL access to SQL features. For example, the UTL\_HTTP package enables HTTP callouts from PL/SQL and SQL to access data on the Internet or to call Oracle Web Server Cartridges. You can use the supplied packages when creating applications or as a source of ideas when creating your own stored procedures. You create a package in two parts: the specification and the body. The package specification declares all public constructs of the package, whereas the package body defines all constructs (public and private) of the package.

## How PL/SQL Runs

PL/SQL supports both native execution and interpreted execution. In interpreted execution, PL/SQL source code is compiled into a so-called bytecode representation. A portable virtual computer implemented as part of Oracle Database runs this bytecode. In native execution, which offers the best performance on computationally intensive program units, the source code of PL/SQL program units is compiled directly to object code for the given platform. This object code is linked into Oracle Database.

The PL/SQL engine is the tool used to define, compile, and run PL/SQL program units. This engine is a special component of many Oracle products, including Oracle Database. While many Oracle products have PL/SQL components, this section specifically covers the program units that can be stored in Oracle Database and processed using Oracle Database PL/SQL engine. The documentation for each Oracle tool describes its PL/SQL capabilities.

Figure 1 illustrates the PL/SQL engine contained in Oracle Database.

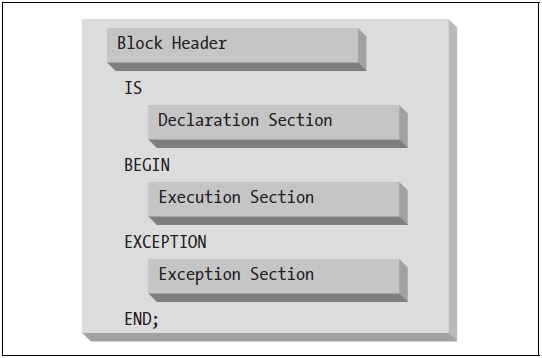


**Figure 1 The PL/SQL Engine and Oracle Database**

The program unit is stored in a database. When an application calls a stored procedure, the database loads the compiled program unit into the shared pool in the system global area (SGA) (see "Shared Pool" on page 14-15). The PL/SQL and SQL statement executors work together to process the statements in the procedure.

You can call a stored procedure from another PL/SQL block, which can be either an anonymous block or another stored procedure. For example, you can call a stored procedure from Oracle Forms.

A PL/SQL procedure executing on Oracle Database can call an external procedure or function written in the C programming language and stored in a shared library. The C routine runs in a separate address space from that of Oracle Database.



**Figure 2 The PL/SQL block structure**

Here is an example of an anonymous block:

DECLARE

today DATE DEFAULT SYSDATE;

BEGIN

-- Display the date.

DBMS\_OUTPUT.PUT\_LINE ('Today is ' || today);

END;

Here is a named block that performs the same action:

CREATE OR REPLACE PROCEDURE show\_the\_date

IS

today DATE DEFAULT SYSDATE;

BEGIN

-- Display the date.

DBMS\_OUTPUT.PUT\_LINE ('Today is ' || today);

END show\_the\_date;

|  |  |
| --- | --- |
| Section | Description |
| Header | Required for named blocks. Specifies the way the program is called by other PL/SQL blocks. Anonymous blocks do not have a header. They start with the DECLARE keyword if there is a declaration section, or with the BEGIN keyword if there are no declarations. |
| Declaration | Optional; declares variables, cursors, TYPEs, and local programs that are used in the block’s execution and exception sections. |
| Execution | Optional in package and TYPE specifications; contains statements that are executed when the block is run. |
| Exception | Optional; describes error-handling behaviour for exceptions raised in the executable section. |

# What is Dynamic SQL?

Most database applications do a specific job. For example, a simple program might prompt the user for an employee number, then update rows in the EMP and DEPT tables. In this case, you know the makeup of the UPDATE statement at precompile time. That is, you know which tables might be changed, the constraints defined for each table and column, which columns might be updated, and the datatype of each column.

However, some applications must accept (or build) and process a variety of SQL statements at run time. For example, a general-purpose report writer must build different SELECT statements for the various reports it generates. In this case, the statement's makeup is unknown until run time. Such statements can, and probably will, change from execution to execution. They are aptly called dynamic SQL statements.

Unlike static SQL statements, dynamic SQL statements are not embedded in your source program. Instead, they are stored in character strings input to or built by the program at run time. They can be entered interactively or read from a file.

## Advantages and Disadvantages of Dynamic SQL

Host programs that accept and process dynamically defined SQL statements are more versatile than plain embedded SQL programs. Dynamic SQL statements can be built interactively with input from users having little or no knowledge of SQL.

For example, your program might simply prompt users for a search condition to be used in the WHERE clause of a SELECT, UPDATE, or DELETE statement. A more complex program might allow users to choose from menus listing SQL operations, table and view names, column names, and so on. Thus, dynamic SQL lets you write highly flexible applications.

However, some dynamic queries require complex coding, the use of special data structures, and more runtime processing. While you might not notice the added processing time, you might find the coding difficult unless you fully understand dynamic SQL concepts and methods.

## When to Use Dynamic SQL

In practice, static SQL will meet nearly all your programming needs. Use dynamic SQL only if you need its open-ended flexibility. Its use is suggested when one of the following items is unknown at precompile time:

* Text of the SQL statement (commands, clauses, and so on)
* The number of host variables
* The datatypes of host variables
* References to database objects such as columns, indexes, sequences, tables, usernames, and views

## PL/SQL examples

Starting from Oracle 8i one can use the "EXECUTE IMMEDIATE" statement to execute dynamic SQL and PL/SQL statements (statements created at run-time). Look at these examples. Note that statements are NOT terminated by semicolons:

EXECUTE IMMEDIATE 'CREATE TABLE x (a NUMBER)';

-- Using bind variables...

sql\_stmt := 'INSERT INTO dept VALUES (:1, :2, :3)';

EXECUTE IMMEDIATE sql\_stmt USING dept\_id, dept\_name, location;

-- Returning a cursor...

sql\_stmt := 'SELECT \* FROM emp WHERE empno = :id';

EXECUTE IMMEDIATE sql\_stmt INTO emp\_rec USING emp\_id;

# Source Books and Articles

1. Kyte T. Expert Oracle Database Architecture: Oracle Database 9i, 10g, and 11g Programming Techniques and Solutions, Second Edition. Apress, 2010.
2. Morton K., & Osborne K., & Sands R., & Shamsudeen R., & Still J. Pro Oracle SQL. Apress, 2013.
3. Powell G. Oracle Data Warehouse Tuning for 10g. Oxford: Elsevier Digital Press, 2005.