PL/SQL

Language Features

- Understanding PL/SQL
- Advantages of PL/SQL
- Performance Advantages
- PL/SQL Program Deployment
- Structure of a PL/SQL Program Block
- Language Syntax

Understanding PL/SQL

- You specify what data is needed. Not how the data is retrieved.
- **Advantage –** Simple
- **Disadvantage –** Occasionally, we need to specify ‘how’

PL/SQL merges the following:

- Logic features of a procedural language (PL)
- Declarative features of SQL

Advantages of PL/SQL

Portability

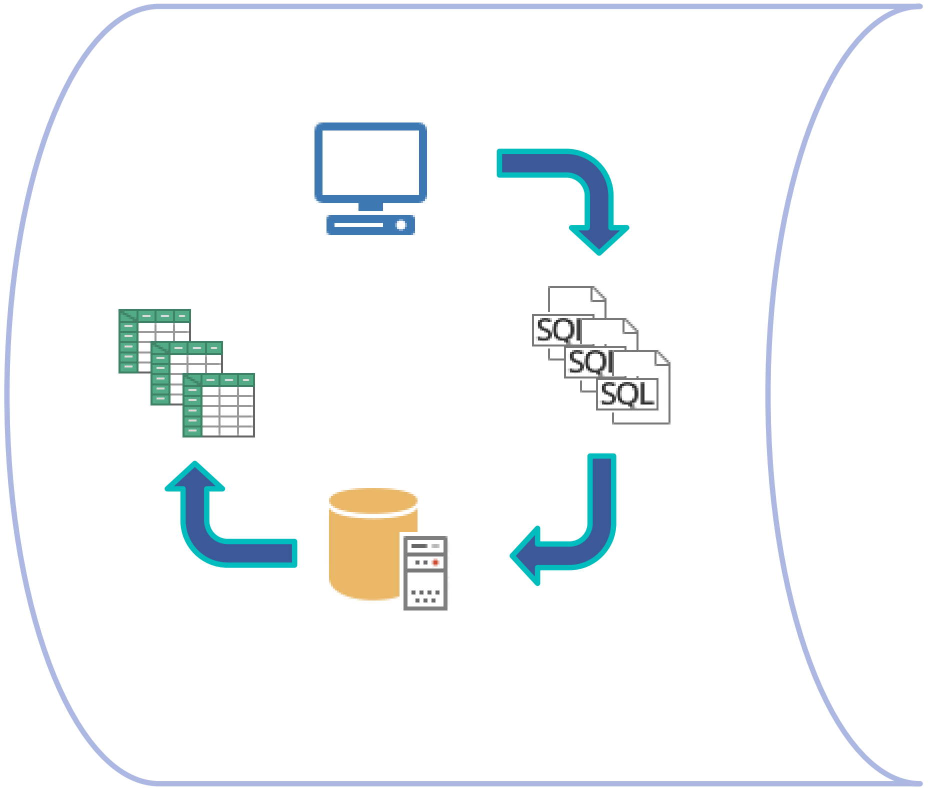
- PL/SQL operates independent of operating system.
- PL/SQL is executed by the Oracle database, not the host system.

Simplicity

- PL/SQL syntax is generally free of complex grammatical rules.

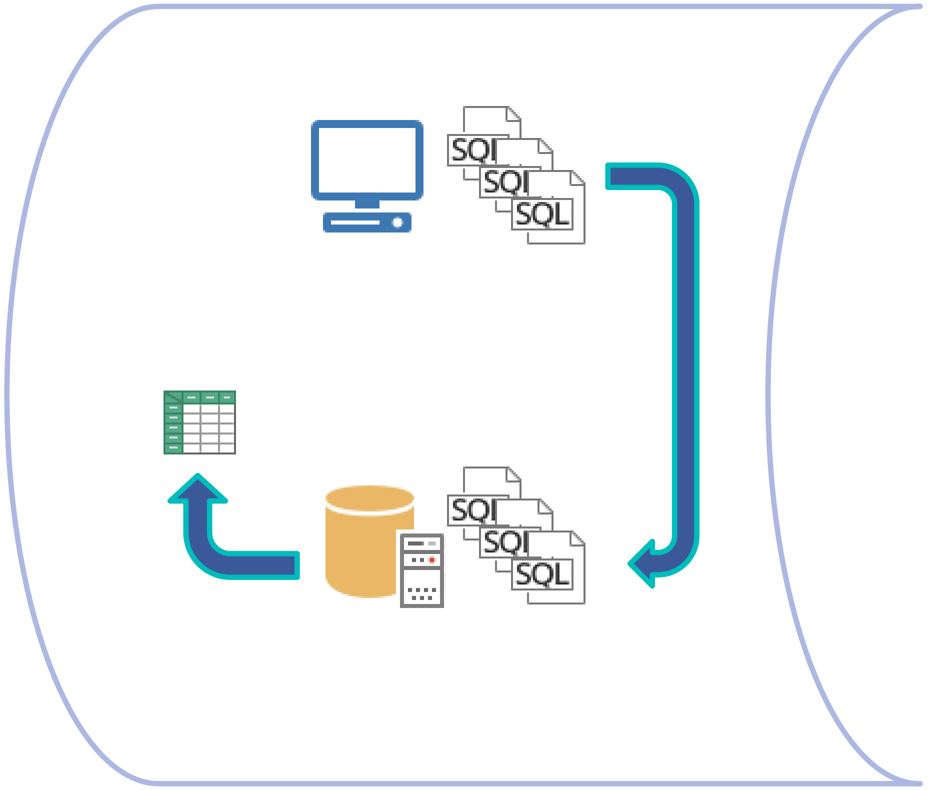
Performance Advantages

Scenario 1 – Without PL/SQL



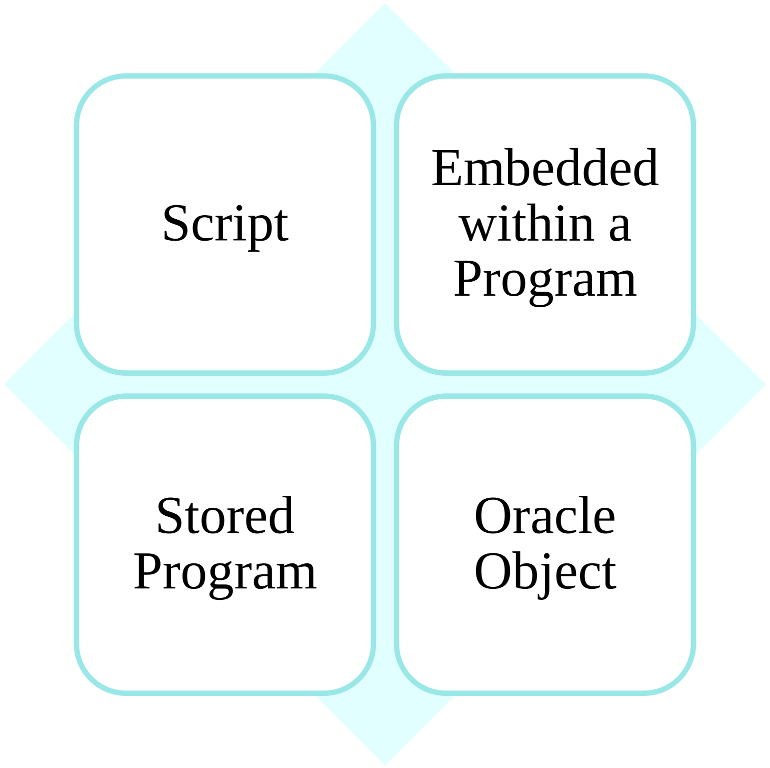
Performance Advantages

Scenario 2 – Using PL/SQL



PL/SQL Program Deployment

There are different situations where you can find PL/SQL programs.



Sent using SQL*Plus client-side session

Such as a server-side trigger.

Script embedded within a C or COBOL program.

A program unit within an Oracle object database

Structure of PL/SQL Block

- PL/SQL instructions are contained within units known as **blocks**.
- Some **blocks** are optional.

Section	Required	Description
DECLARE		Declares internal program objects, such as variables.
BEGIN	☑	Marks the beginning of the program logic.
Program Logic	☑	This is the actual PL/SQL and SQL statements.
EXCEPTION		Marks the beginning of exception logic.
END	☑	Marks the end of the program logic.

Language Syntax Rules

- Commenting Code
  - Like any programming language, you need to be able to comment your code.
- **In Line** comment markers ‘- -’

```
BEGIN
-- Populate a table with 100 rows of test data.
     FOR I IN 1..1000 LOOP
     -- Include SQL DML statement
          INSERT INTO employee (ssn, name)
          VALUES (900000000 + I, ‘John Doe’);
…
```

Language Syntax Rules

- **Multi Line** comment markers consist of
  - beginning marker (/*)
  - end marker (*/)

```
/*
Block comment outlined by the begin and end marks
*/
BEGIN
-- Populate a table with 100 rows of test data.
       FOR I IN 1..1000 LOOP
              INSERT INTO employee (ssn, name)
              VALUES (900000000 + I, ‘John Doe’);
```

Language Syntax Rules

- When writing PL/SQL code, remember the following:
  - Only one PL/SQL statement per line
  - All **execution** statement must be terminated with a semi-colon (;)

```
BEGIN
        FOR I IN 1..1000 LOOP
                INSERT INTO employee (ssn, name)
                VALUES (900000000 + I, 'John Doe');
        END LOOP;
…
```

Language Syntax Rules

– Statements that simply label a portion of PL/SQL code are **not** terminated with the semi-colon.

```
BEGIN
        FOR I IN 1..1000 LOOP
                INSERT INTO employee (ssn, name)
                VALUES (900000000 + I, ‘John Doe’);
        END LOOP;
…
```

Language Syntax Rules Overview

```
DECLARE

BEGIN
        FO          IN 1..1000 LOOP
                INSERT INTO employee (ssn, name)
                VALUES (900000000 + I, 'John Doe');
                  OOP;
        COMMIT;

EXCEPTION
        WHEN OTHERS THEN
                ROLLBACK;

END;
/
```

This is optional, but kept for completeness

This is a standard SQL statement within PL/SQL Loop

The COMMIT statement executes after the loop

In the event of an error, all prior execution is rolled back

See it in Action

Lab Exercises

- Display salary of employee against ssn provided by a user.
- Identify the one employee who has been the least active, based upon the number of hours they have been working on projects. This will be the first employee we want to remove from the existing COMPANY database and transfer them into the new division.