

# What Are PL/SQL Packages?

- A package is a schema object that groups logically related PL/SQL types, variables, and subprograms.
- Packages usually have two parts:
  - A specification (spec)
  - A body
- The specification is the interface to the package. It declares the types, variables, constants, exceptions, cursors, and subprograms that can be referenced from outside the package.
- The body defines the queries for the cursors and the code for the subprograms.
- Enable the Oracle server to read multiple objects into memory at once.

ORACLE

12-1

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

## PL/SQL Packages: Overview

PL/SQL packages enable you to bundle related PL/SQL types, variables, data structures, exceptions, and subprograms into one container. For example, a Human Resources package can contain hiring and firing procedures, commission and bonus functions, and tax exemption variables.

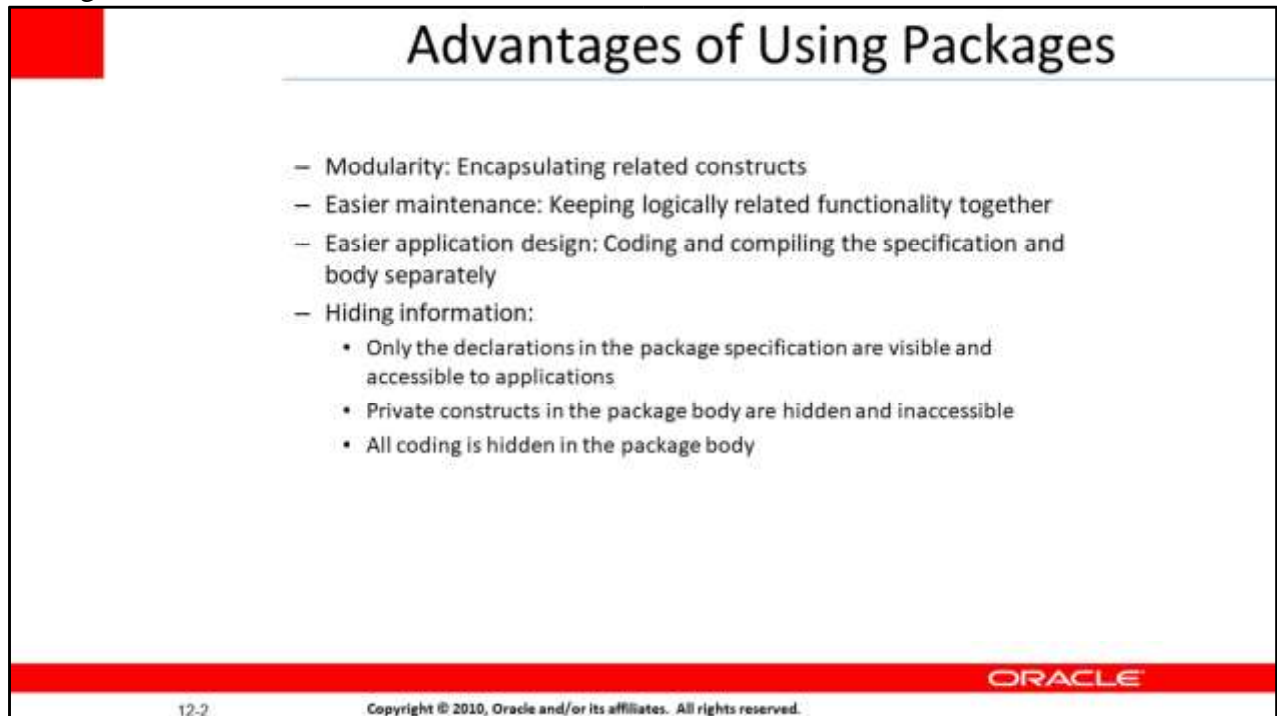
A package usually consists of two parts stored separately in the database:

- A specification

- A body (optional)

The package itself cannot be called, parameterized, or nested. After writing and compiling, the contents can be shared with many applications.

When a PL/SQL-packaged construct is referenced for the first time, the whole package is loaded into memory. Subsequent access to constructs in the same package does not require disk input/output (I/O).



**Advantages of Using Packages**

- Modularity: Encapsulating related constructs
- Easier maintenance: Keeping logically related functionality together
- Easier application design: Coding and compiling the specification and body separately
- Hiding information:
  - Only the declarations in the package specification are visible and accessible to applications
  - Private constructs in the package body are hidden and inaccessible
  - All coding is hidden in the package body

12-2 Copyright © 2010, Oracle and/or its affiliates. All rights reserved. ORACLE

### Advantages of Using Packages

Packages provide an alternative to creating procedures and functions as standalone schema objects, and they offer several benefits.

**Modularity and easier maintenance:** You encapsulate logically related programming structures in a named module. Each package is easy to understand, and the interface between packages is simple, clear, and well defined.

**Easier application design:** All you need initially is the interface information in the package specification. You can code and compile a specification without its body. Thereafter, stored subprograms that reference the package can compile as well. You need not define the package body fully until you are ready to complete the application.

**Hiding information:** You decide which constructs are public (visible and accessible) and which are private (hidden and inaccessible). Declarations in the package specification are visible and accessible to applications. The package body hides the definition of the private constructs, so that only the package is affected (not your application or any calling programs) if the definition changes. This enables you to change the implementation without having to recompile the calling programs. Also, by hiding implementation

Program Units 3 - 2

details from users, you protect the integrity of the package.



## Advantages of Using Packages

- Added functionality: Persistency of public variables and cursors
- Better performance:
  - The entire package is loaded into memory when the package is first referenced.
  - There is only one copy in memory for all users.
  - The dependency hierarchy is simplified.
- Overloading: Multiple subprograms of the same name

ORACLE

12-3

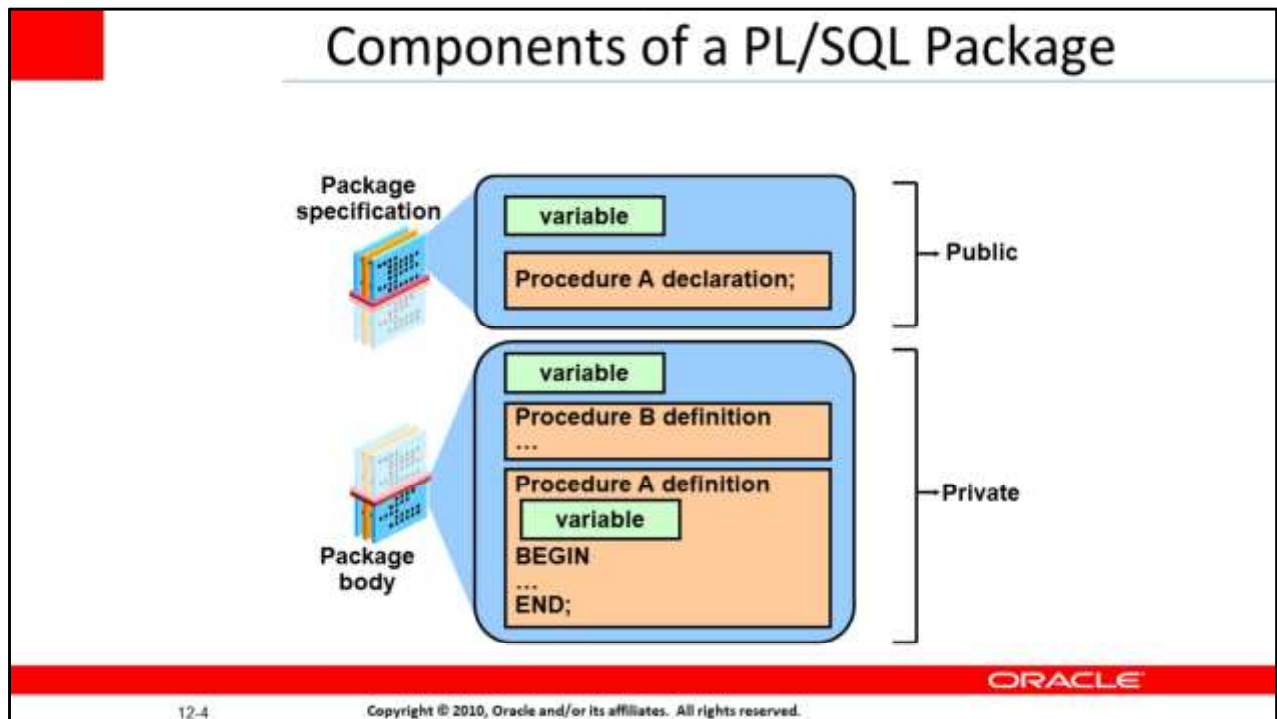
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

### Advantages of Using Packages (continued)

**Added functionality:** Packaged public variables and cursors persist for the duration of a session. Thus, they can be shared by all subprograms that execute in the environment. They also enable you to maintain data across transactions without having to store it in the database. Private constructs also persist for the duration of the session but can be accessed only within the package.

**Better performance:** When you call a packaged subprogram the first time, the entire package is loaded into memory. Later calls to related subprograms in the package, therefore, require no further disk I/O. Packaged subprograms also stop cascading dependencies and thus avoid unnecessary compilation. **Overloading:** With packages, you can overload procedures and functions, which means you can create multiple subprograms with the same name in the same package, each taking parameters of different number or data type.

**Note:** Dependencies are covered in detail in the lesson titled “Managing Dependencies.”

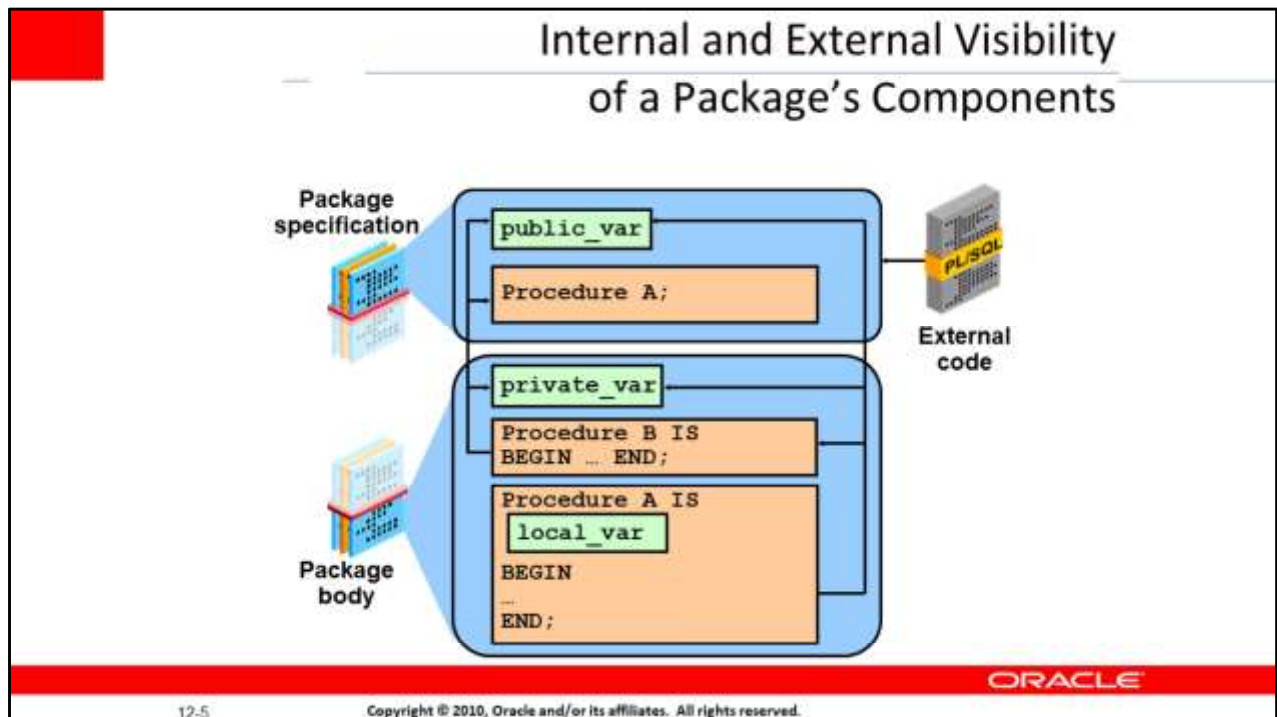


### Components of a PL/SQL Package

You create a package in two parts:

The package specification is the interface to your applications. It declares the public types, variables, constants, exceptions, cursors, and subprograms available for use. The package specification may also include PRAGMAs, which are directives to the compiler. The package body defines its own subprograms and must fully implement subprograms declared in the specification part. The package body may also define PL/SQL constructs, such as types, variables, constants, exceptions, and cursors.

Public components are declared in the package specification. The specification defines a public application programming interface (API) for users of package features and functionality—that is, public components can be referenced from any Oracle server environment that is external to the package. Private components are placed in the package body and can be referenced only by other constructs within the same package body. Private components can reference the public components of a package.



Note: If a package specification does not contain subprogram declarations, then there is no requirement for a package body.

### Internal and External Visibility of a Package's Components

The visibility of a component describes whether that component can be seen, that is, referenced and used by other components or objects. The visibility of components depends on whether they are locally or globally declared. Local components are visible within the structure in which they are declared, such as:

Variables defined in a subprogram can be referenced within that subprogram, and are not visible to external components—for example, `local_var` can be used in procedure A.

Private package variables, which are declared in a package body, can be referenced by other components in the same package body. They are not visible to any subprograms or objects that are outside the package. For example, `private_var` can be used by procedures A and B within the package body, but not outside the package.

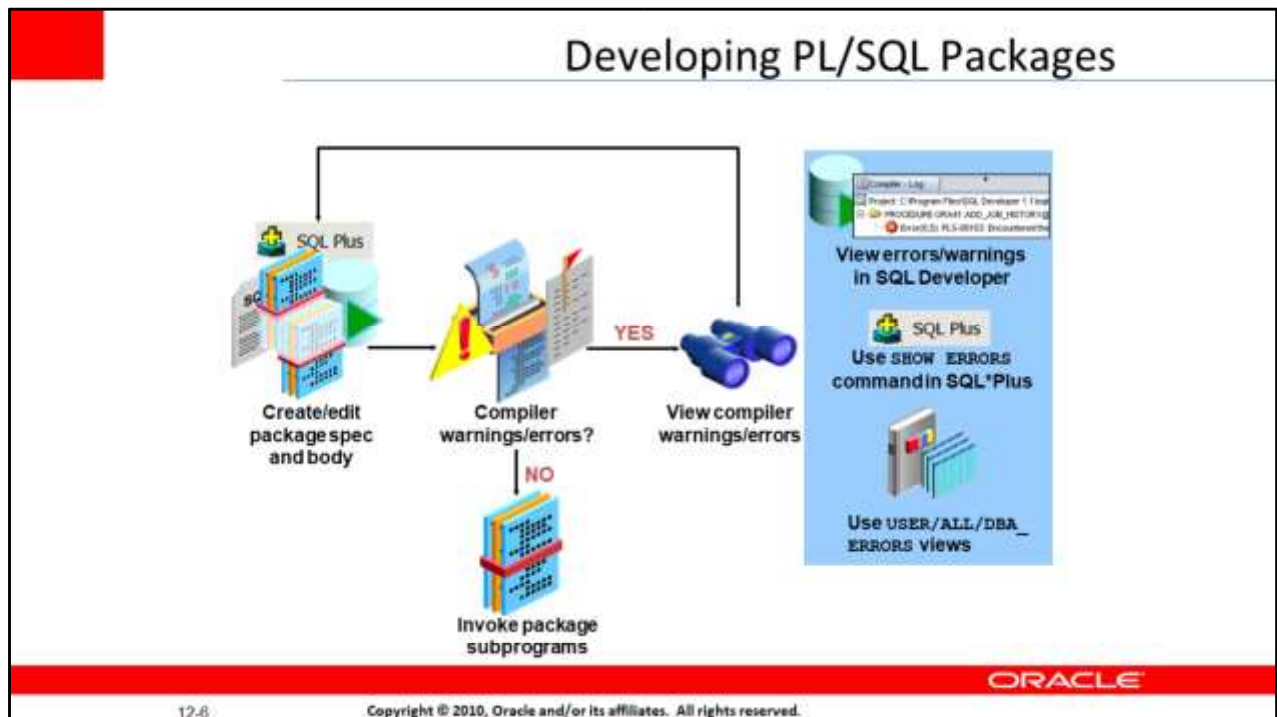
Globally declared components are visible internally and externally to the package, such as:

A public variable, which is declared in a package specification, can be referenced and changed outside the package (for example, `public_var` can be referenced externally).



A package subprogram in the specification can be called from external code sources (for example, procedure A can be called from an environment external to the package).

Note: Private subprograms, such as procedure B, can be invoked only with public subprograms, such as procedure A, or other private package constructs. A public variable declared in the package specification is a global variable.



### Developing PL/SQL Packages

The graphic in the slide illustrates the basic steps involved in developing and using a package:

1. Create the procedure using SQL Developer's Object Navigator tree or the SQL Worksheet area.
2. Compile the package. The package is created in the database. The CREATEPACKAGE statement creates and stores source code and the compiled m-code in the database. To compile the package, right-click the package's name in the Object Navigator tree, and then click Compile.
3. If there are no compilation warnings or errors, you execute any public construct within the package specification from an Oracle Server environment.
4. If there are compilation warning or errors, you can view (and then correct) the warnings or errors using one of the following methods:

Using the SQL Developer interface (the Compiler – Log tab)  
 Using the SHOW ERRORS SQL\*Plus command  
 Using the USER/ALL/DBA\_ERRORS views

Program Units 3 - 6