12

Creating Packages

ORACLE

Objectives

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

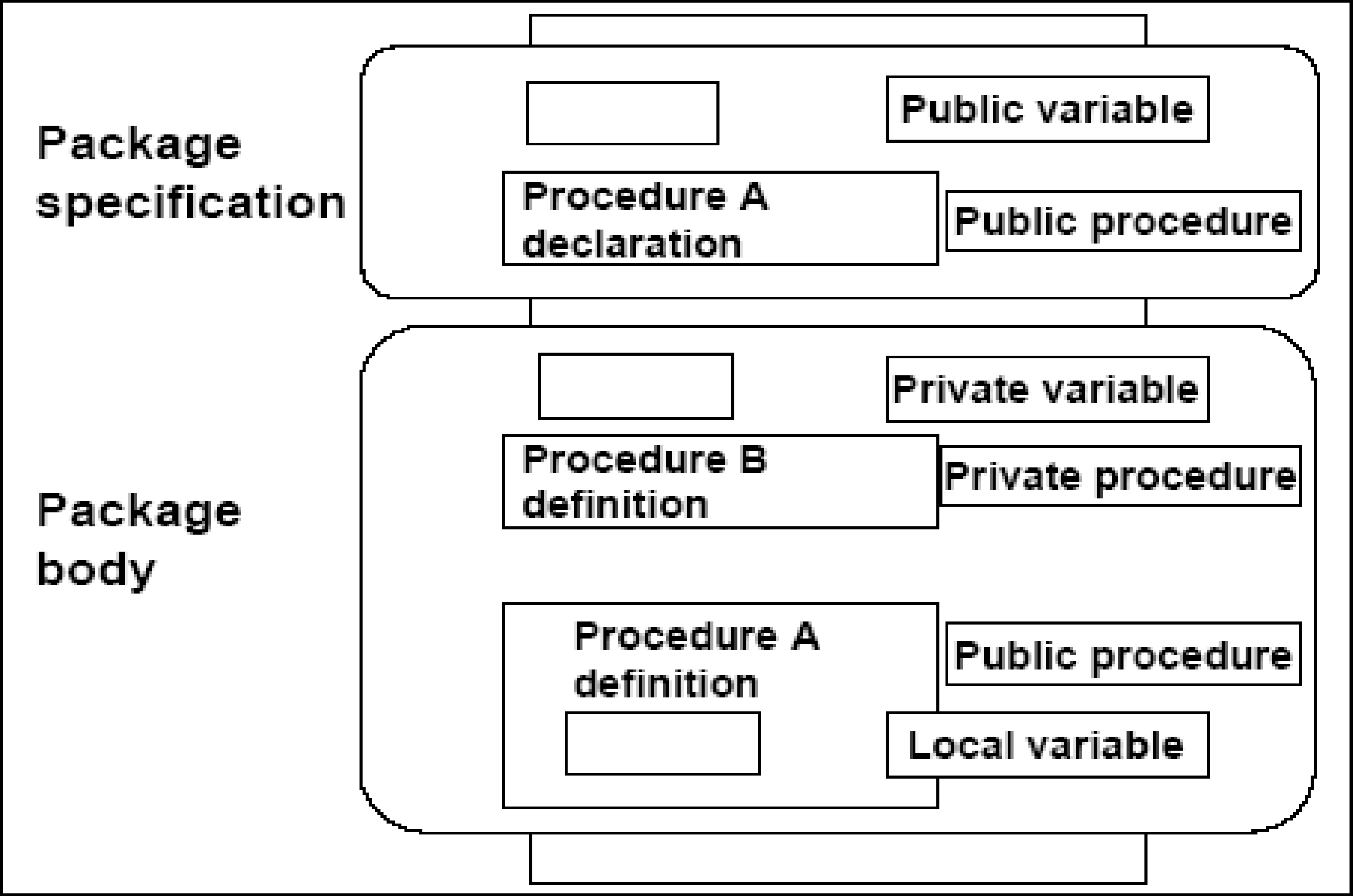
- Describe packages and list their possible components
- Create a package to group together related variables, cursors, constants, exceptions, procedures, and functions
- Designate a package construct as either public or private
- Invoke a package construct
- Describe a use for a bodiless package

Overview of Packages

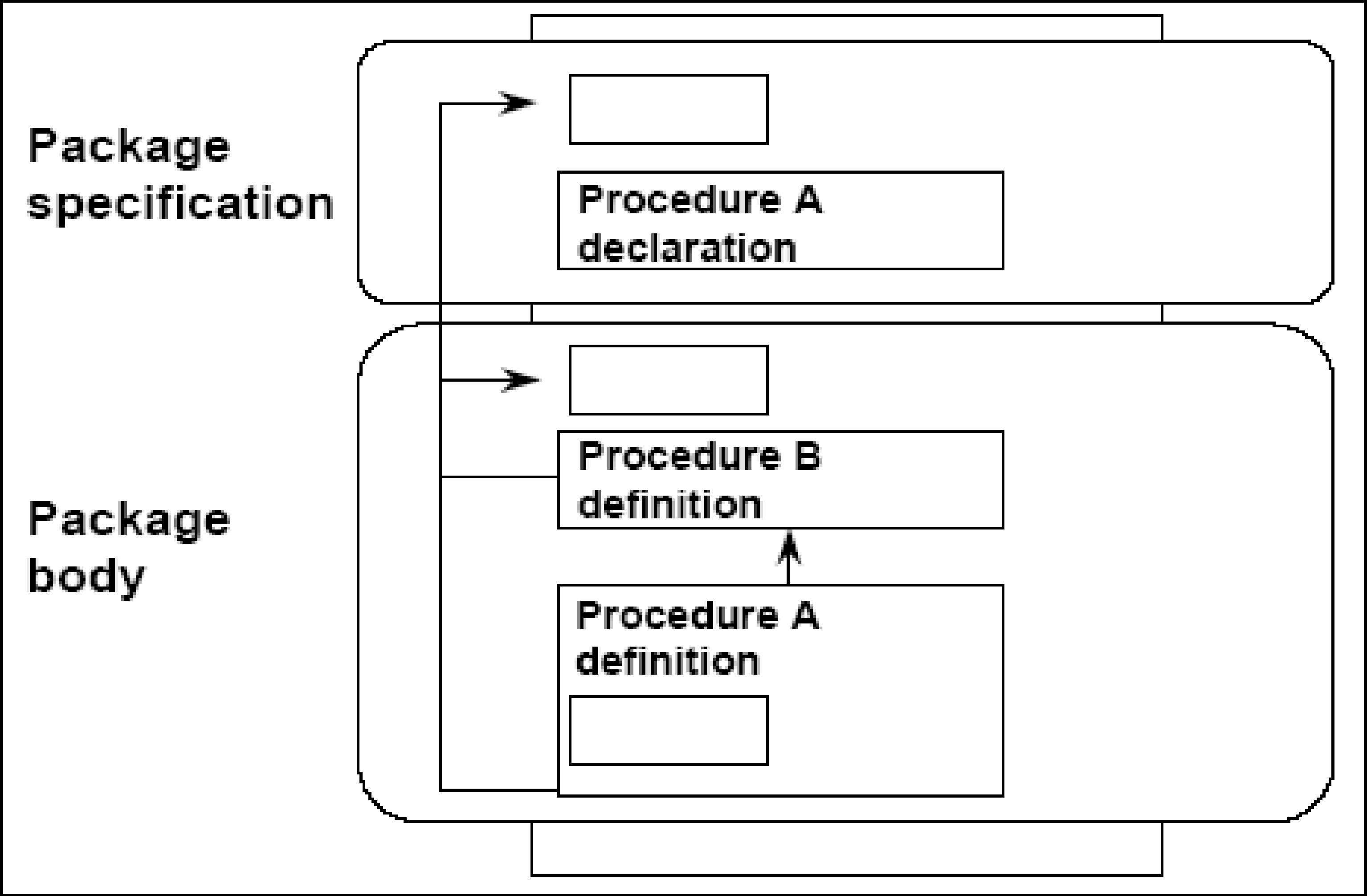
Packages:

- Group logically related PL/SQL types, items, and subprograms
- Consist of two parts:
  - Specification
  - Body
- Cannot be invoked, parameterized, or nested
- Allow the Oracle server to read multiple objects into memory at once

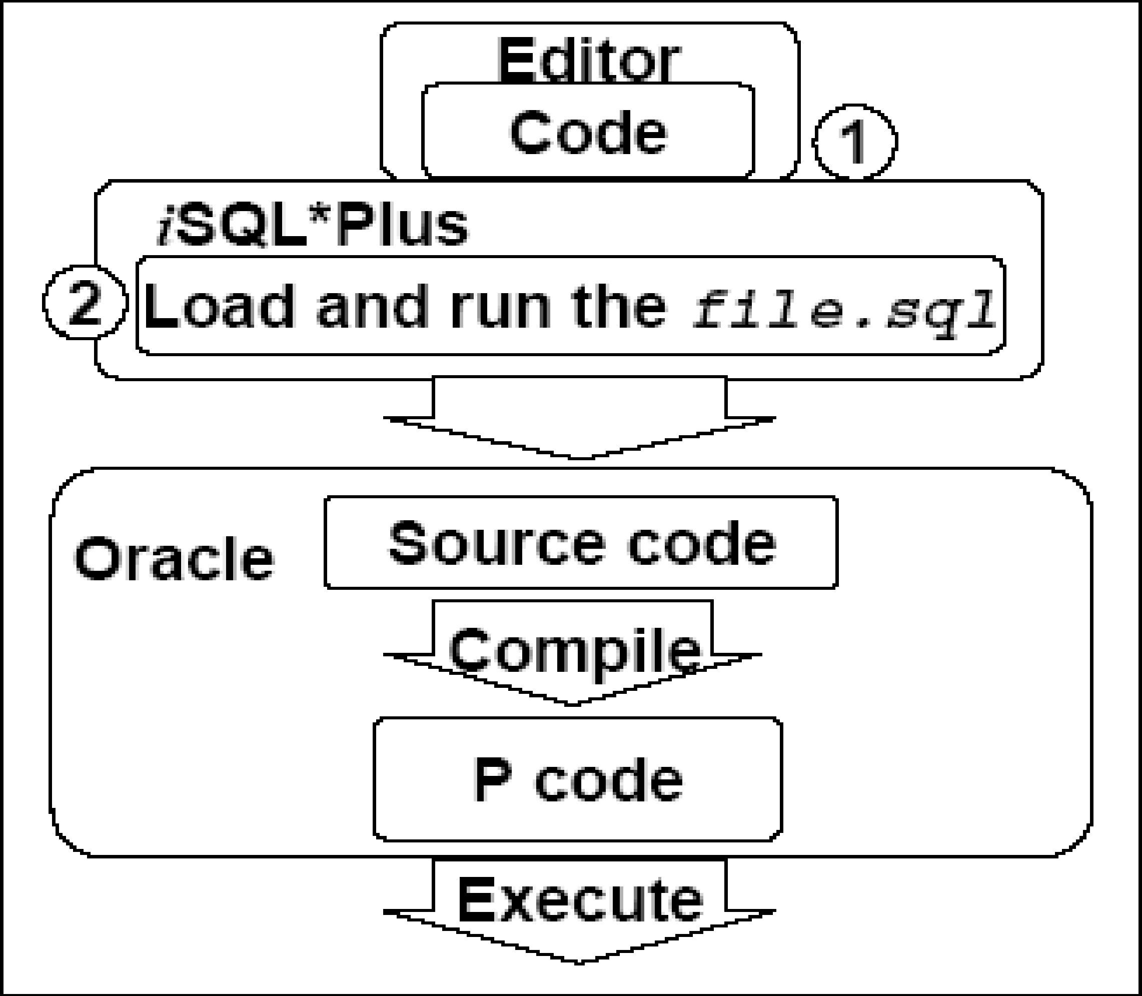
Components of a Package



Referencing Package Objects



Developing a Package



Developing a Package

- Saving the text of the CREATE PACKAGE statement in two different SQL files facilitates later modifications to the package.
- A package specification can exist without a package body, but a package body cannot exist without a package specification.

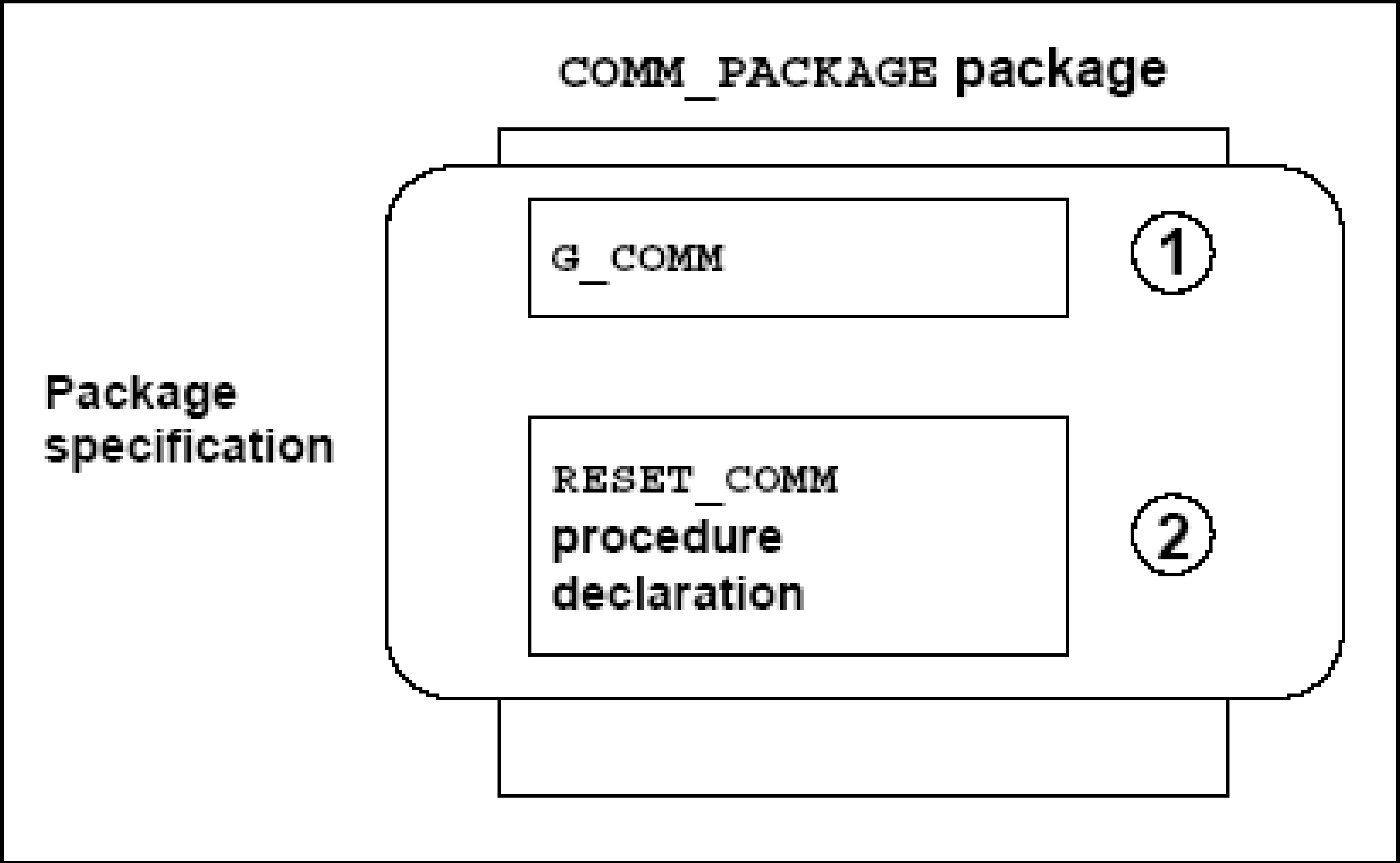
Creating the Package Specification

Syntax:

```
CREATE  [OR REPLACE]  PACKAGE  package_name
IS|AS
                public type and item declarations
                subprogram specifications
END  package_name;
```

- The REPLACE option drops and recreates the package specification.
- Variables declared in the package specification are initialized to NULL by default.
- All the constructs declared in a package specification are visible to users who are granted privileges on the package.

Declaring Public Constructs



Creating a Package Specification: Example

```
CREATE OR REPLACE PACKAGE comm_package IS
     g_comm   NUMBER  :=  0.10;          --initialized to 0.10
     PROCEDURE reset_comm
          (p_comm IN NUMBER);
END comm_package;
/
```

```
Package created
```

- G_COMM is a global variable and is initialized to 0.10.
- RESET_COMM is a public procedure that is implemented in the package body.

ORACLE

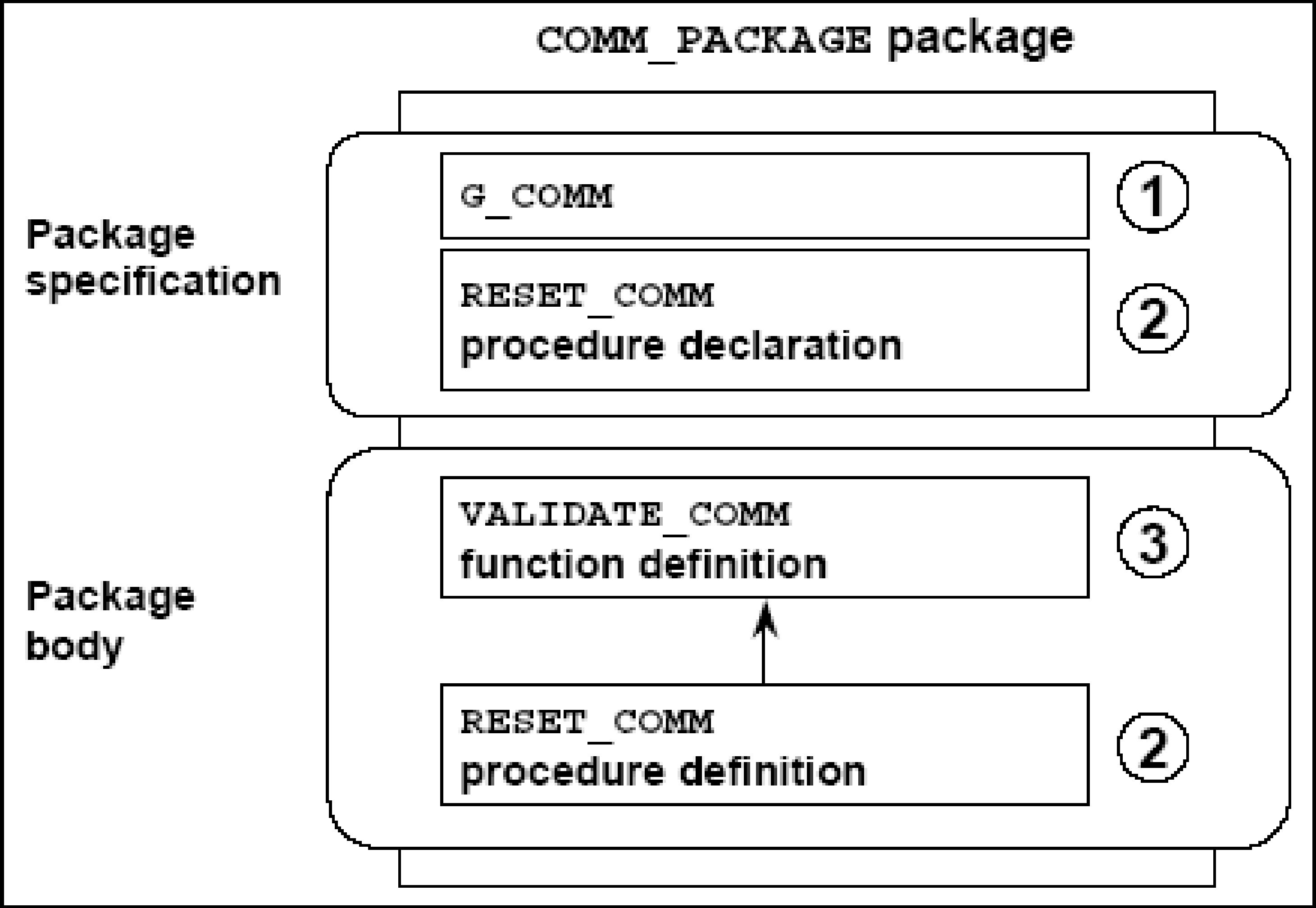
Creating the Package Body

Syntax:

```
CREATE  [OR REPLACE]  PACKAGE BODY  package_name
IS|AS
                private type and item declarations
                subprogram bodies
END  package_name;
```

- The REPLACE option drops and recreates the package body.
- Identifiers defined only in the package body are private constructs. These are not visible outside the package body.
- All private constructs must be declared before they are used in the public constructs.

Public and Private Constructs


COMM_PACKAGE package
Package specification
G_COMM
1
RESET_COMM procedure declaration
2
Package body
VALIDATE_COMM function definition
3
RESET_COMM procedure definition
2

Creating a Package Body: Example

comm_pack.sql

```
CREATE OR REPLACE PACKAGE BODY comm_package
IS
   FUNCTION validate_comm (p_comm IN NUMBER)
   RETURN BOOLEAN
   IS
            v_max_comm NUMBER;
BEGIN
     SELECT MAX(commission_pct)
        INTO v_max_comm
        FROM employees;
     IF p_comm > v_max_comm THEN RETURN(FALSE);
     ELSE RETURN(TRUE);
     END IF;
END validate_comm;
...
```

Creating a Package Body: Example

comm_pack.sql

```
PROCEDURE reset_comm (p_comm IN NUMBER)
IS
BEGIN
  IF validate_comm(p_comm)  THEN
      g_comm:=p_comm; --reset global variable
  ELSE
     RAISE_APPLICATION_ERROR(-20210,'Invalid commission');
  END IF;
END reset_comm;
END comm_package;
/
```

Package body created

Invoking Package Constructs

Example 1: Invoke a function from a procedure within the same package.

```
CREATE OR REPLACE PACKAGE BODY comm_package IS
    . . .
  PROCEDURE reset_comm (p_comm IN NUMBER)
  IS
  BEGIN
              IF validate_comm(p_comm) THEN
                  g_comm := p_comm;
              ELSE
                 RAISE_APPLICATION_ERROR
                              (-20210, 'Invalid commission');
           END IF;
  END reset_comm;
END comm_package;
```

Invoking Package Constructs

Example 2: Invoke a package procedure from *i*SQL*Plus.

```
EXECUTE comm_package.reset_comm(0.15)
```

Example 3: Invoke a package procedure in a different schema.

```
EXECUTE scott.comm_package.reset_comm(0.15)
```

Example 4: Invoke a package procedure in a remote database.

```
EXECUTE comm_package.reset_comm@ny(0.15)
```

Declaring a Bodiless Package

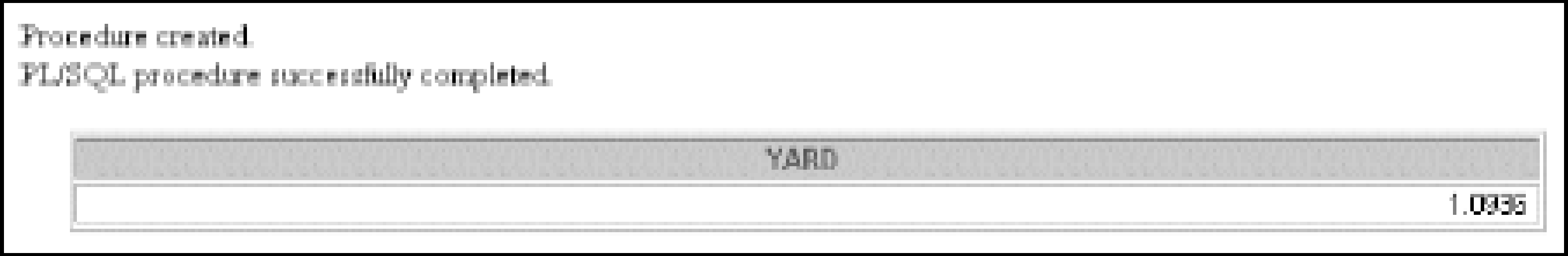
```
SET SERVEROUTPUT ON
CREATE OR REPLACE PACKAGE global_consts IS
   mile_2_kilo     CONSTANT  NUMBER  := 1.6093;
   kilo_2_mile     CONSTANT  NUMBER  := 0.6214;
   yard_2_meter    CONSTANT  NUMBER  := 0.9144;
   meter_2_yard    CONSTANT  NUMBER  := 1.0936;
END global_consts;
/
EXECUTE  DBMS_OUTPUT.PUT_LINE ('20 miles = '||20*
                              global_consts.mile_2_kilo||' km')
```

```
Package created
20 miles = 32.186 km
PL/SQL procedure successfully completed.
```

Referencing a Public Variable from a Stand-Alone Procedure

Example:

```
CREATE OR REPLACE PROCEDURE meter_to_yard
              (p_meter IN NUMBER, p_yard OUT NUMBER)
IS
BEGIN
   p_yard := p_meter * global_consts.meter_2_yard;
END meter_to_yard;
/
VARIABLE yard NUMBER
EXECUTE meter_to_yard (1, :yard)
PRINT yard
```



```
Procedure created.
PL/SQL procedure successfully completed.
```

YARD
1.0936

Removing Packages

To remove the package specification and the body, use the following syntax:

```
DROP PACKAGE package_name;
```

To remove the package body, use the following syntax:

```
DROP PACKAGE BODY package_name;
```

Guidelines for Developing Packages

- Construct packages for general use.
- Define the package specification before the body.
- The package specification should contain only those constructs that you want to be public.
- Place items in the declaration part of the package body when you must maintain them throughout a session or across transactions.
- Changes to the package specification require recompilation of each referencing subprogram.
- The package specification should contain as few constructs as possible.

Advantages of Packages

- Modularity: Encapsulate related constructs.
- Easier application design: Code and compile 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.

ORACLE

Advantages of Packages

- Added functionality: Persistency of 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

Summary

In this lesson, you should have learned how to:

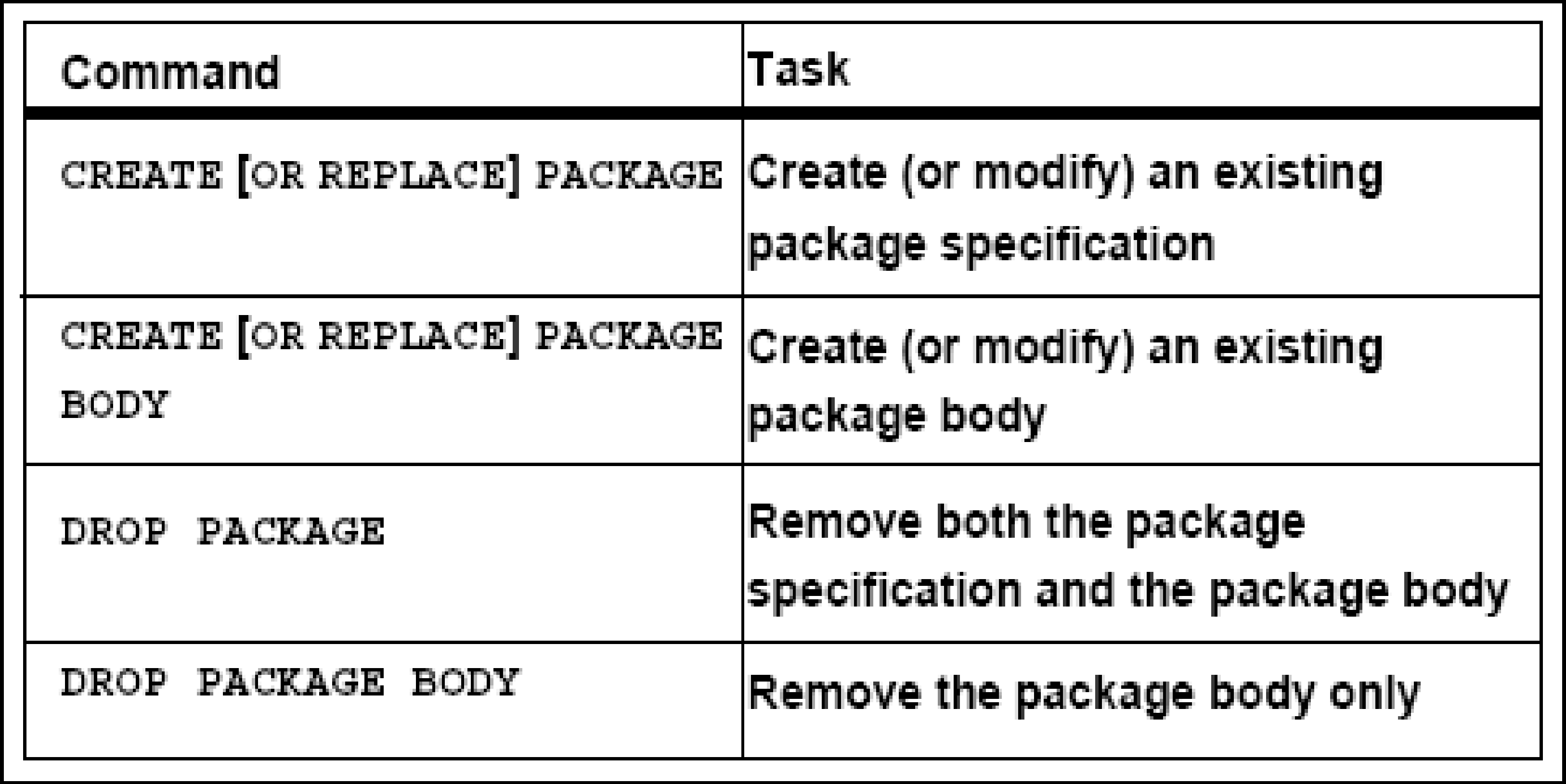
- Improve organization, management, security, and performance by using packages
- Group related procedures and functions together in a package
- Change a package body without affecting a package specification
- Grant security access to the entire package

Summary

In this lesson, you should have learned how to:

- Hide the source code from users
- Load the entire package into memory on the first call
- Reduce disk access for subsequent calls
- Provide identifiers for the user session

Summary



Command	Task
`CREATE [OR REPLACE] PACKAGE`	Create (or modify) an existing package specification
`CREATE [OR REPLACE] PACKAGE BODY`	Create (or modify) an existing package body
`DROP PACKAGE`	Remove both the package specification and the package body
`DROP PACKAGE BODY`	Remove the package body only

Practice 12 Overview

This practice covers the following topics:

- Creating packages
- Invoking package program units