

Creating the Package Specification: Using the CREATE PACKAGE Statement

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

- The OR REPLACE option drops and re-creates 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.

ORACLE

12-1

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

Creating the Package Specification

To create packages, you declare all public constructs within the package specification.

Specify the ORREPLACE option if overwriting an existing package specification.

Initialize a variable with a constant value or formula within the declaration, if required; otherwise, the variable is initialized implicitly to NULL.

The following are definitions of items in the package syntax:

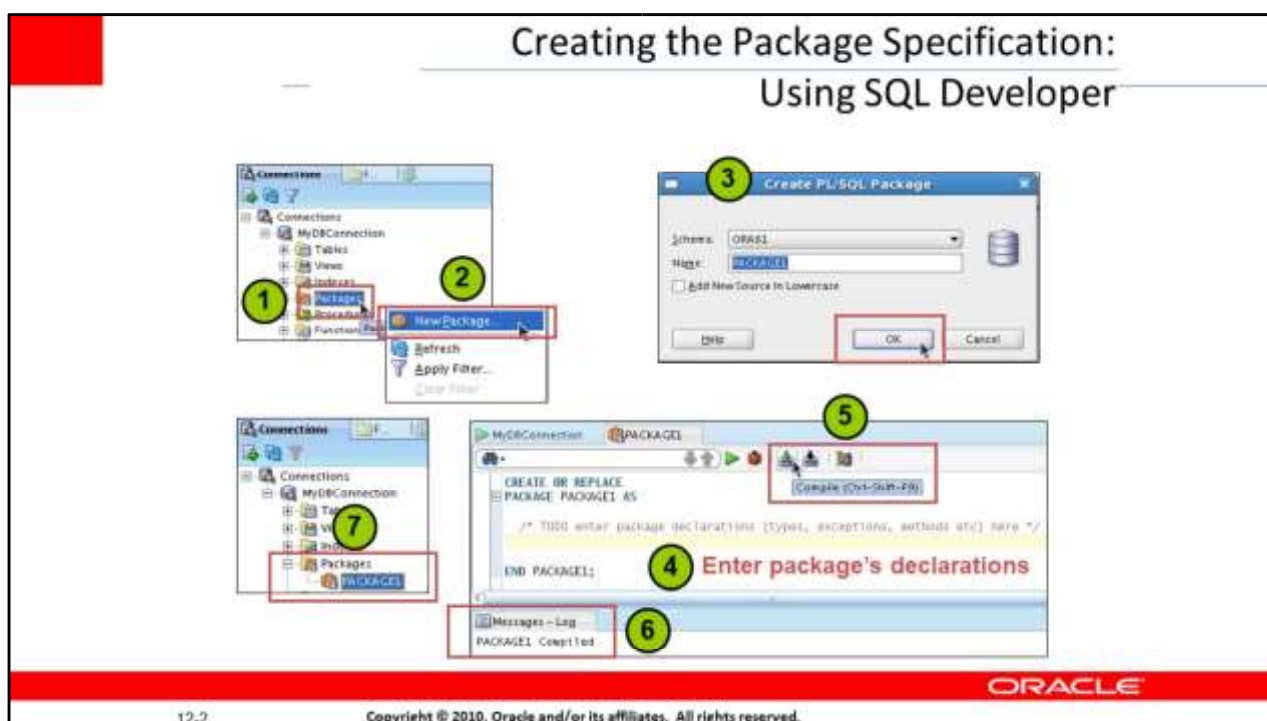
- *package_name* specifies a name for the package that must be unique among objects within the owning schema. Including the package name after the END keyword is optional.
- *public type and variable declarations* declares public variables, constants, cursors, exceptions, user-defined types, and subtypes.
- *subprogram specification* specifies the public procedure or function declarations.

The package specification should contain procedure and function headings terminated by a semicolon, without the IS (or AS) keyword and its PL/SQL

Program Units 3 - 1

block. The implementation of a procedure or function that is declared in a package specification is done in the package body.

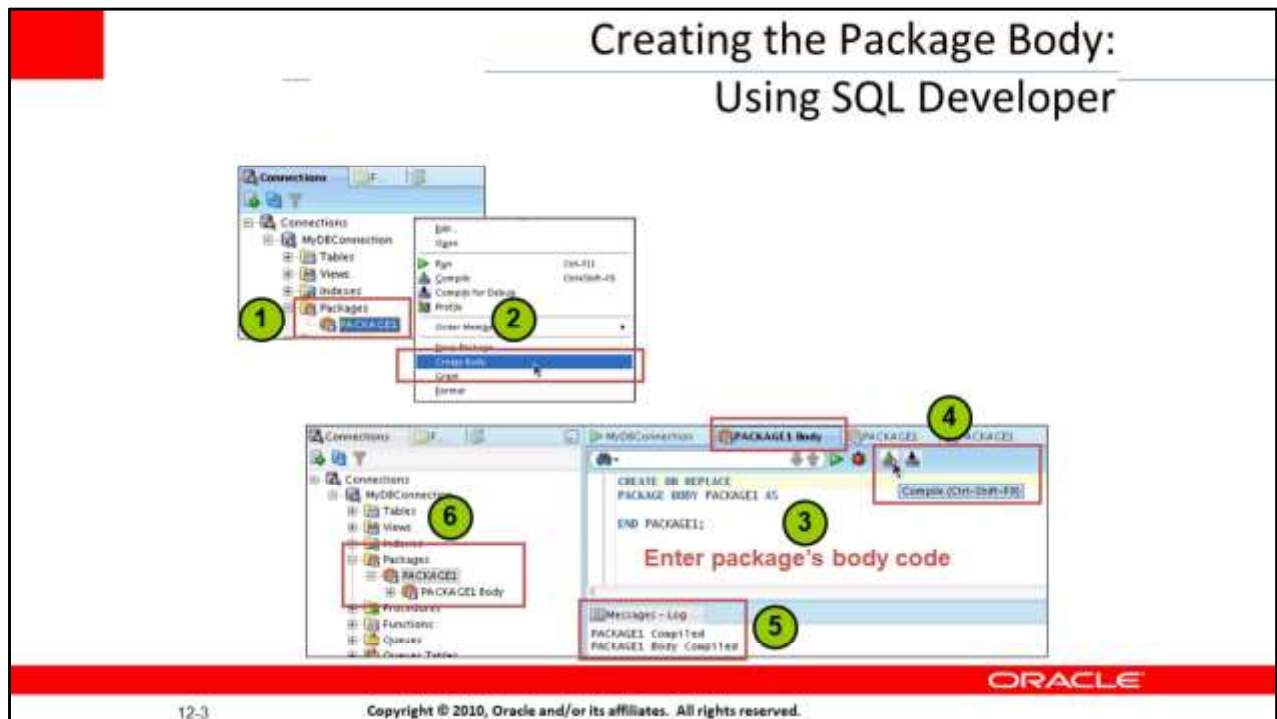
The Oracle database stores the specification and body of a package separately. This enables you to change the implementation of a program construct in the package body without invalidating other schema objects that call or reference the program construct.



Creating the Package Specification: Using SQL Developer

You can use SQL Developer to create the package specification as follows:

1. Right-click the Packages node in the Connections navigation tree.
2. Select New Package from the shortcut menu.
3. In the Create PL/SQL Package window, select the schema name, enter the name for the new package, and then click OK. A tab for the new package is displayed along with the shell for the new package.
4. Enter the code for the new package.
5. Compile or save (using the Save icon on the main toolbar) the new package.
6. The Messages – Log tab displays whether or not the compilation was successful.
7. The newly created package is displayed under the Packages node in the Connections navigation tree.



Creating the Package Body: Using SQL Developer

- You can use SQL Developer to create the package body as follows:
1. Right-click the package name for which you are creating a body in the Packages node in the Connections navigation tree.
 2. Select Create Body from the shortcut menu. A tab for the new package body is displayed along with the shell for the new package body.
 3. Enter the code for the new package body.
 4. Compile or save the package body.
 5. The Messages – Log tab displays whether or not the compilation was successful.
 6. The newly created package body is displayed under the Packages node in the Connections navigation tree.

Example of Package Specification: comm_pkg

The example in the slide creates a package called comm_pkg used to manage business processing rules for commission calculations.

Creating the Package Body

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

- The OR REPLACE option drops and re-creates the package body.
- Identifiers defined in the package body are *private* and not visible outside the package body.
- All *private* constructs must be declared before they are referenced.
- Public constructs are visible to the package body.

ORACLE

12-5

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

The v_std_comm public (global) variable is declared to hold a maximum allowable percentage commission for the user session, and it is initialized to 0.10 (that is, 10%).

The reset_comm public procedure is declared to accept a new commission percentage that updates the standard commission percentage if the commission validation rules are accepted. The validation rules for resetting the commission are not made public and do not appear in the package specification. The validation rules are managed by using a private function in the package body.

Creating the Package Body

Create a package body to define and implement all public subprograms and supporting private constructs. When creating a package body, perform the following steps:

Specify the ORREPLACE option to overwrite an existing package body.

Define the subprograms in an appropriate order. The basic principle is that you must declare a variable or subprogram before it can be referenced by other components in the same package body. It is

common to see all private variables and subprograms defined first and the public subprograms defined last in the package body.

Complete the implementation for all procedures or functions declared in the package specification within the package body.

The following are definitions of items in the package body syntax:

- package_name specifies a name for the package that must be the same as its package specification. Using the package name after the END keyword is optional.
- private type and variable declarations declares private variables, constants, cursors, exceptions, user-defined types,

and subtypes.

- subprogram specification specifies the full implementation of any private and/or public procedures or functions.

- [BEGIN initialization statements] is an optional block of initialization code that executes when the package is first referenced.

Example of a Package Body: comm_pkg

```
CREATE OR REPLACE PACKAGE BODY comm_pkg IS
  FUNCTION validate(p_comm NUMBER) RETURN BOOLEAN IS
    v_max_comm employees.commission_pct%type;
  BEGIN
    SELECT MAX(commission_pct) INTO v_max_comm
    FROM employees;
    RETURN (p_comm BETWEEN 0.0 AND v_max_comm);
  END validate;

  PROCEDURE reset_comm (p_new_comm NUMBER) IS
  BEGIN
    IF validate(p_new_comm) THEN
      v_std_comm := p_new_comm; -- reset public var
    ELSE RAISE_APPLICATION_ERROR(
      -20210, 'Bad Commission');
    END IF;
  END reset_comm;
END comm_pkg;
```

ORACLE

12-6

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

Example of a Package Body: comm_pkg

The slide shows the complete package body for comm_pkg, with a private function called validate to check for a valid commission. The validation requires that the commission be positive and less than the highest commission among existing employees. The reset_comm procedure invokes the private validation function before changing the standard commission in v_std_comm. In the example, note the following:

- The v_std_comm variable referenced in the reset_comm procedure is a public variable. Variables declared in the package specification, such as v_std_comm, can be directly referenced without qualification.
- The reset_comm procedure implements the public definition in the specification.
- In the comm_pkg body, the validate function is private and is directly referenced from the reset_comm procedure without qualification.

Note: The validate function appears before the reset_comm procedure because the reset_comm procedure references the validate function. It is possible to create forward declarations for subprograms in the package body if

Program Units 3 - 6

their order of appearance needs to be changed. If a package specification declares only types, constants, variables, and exceptions without any subprogram specifications, then the package body is unnecessary. However, the body can be used to initialize items declared in the package specification.

Invoking the Package Subprograms

```
-- Invoke a function within the same packages:
CREATE OR REPLACE PACKAGE BODY comm_pkg IS ...
  PROCEDURE reset_comm(p_new_comm NUMBER) IS
  BEGIN
    IF validate(p_new_comm) THEN
      v_std_comm := p_new_comm;
    ELSE ...
    END IF;
  END reset_comm;
END comm_pkg;
```

```
-- Invoke a package procedure from SQL*Plus:
EXECUTE comm_pkg.reset_comm(0.15)
```

```
-- Invoke a package procedure in a different schema:
EXECUTE scott.comm_pkg.reset_comm(0.15)
```

ORACLE

12-7

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

Invoking Package Subprograms

After the package is stored in the database, you can invoke public or private subprograms within the same package, or public subprograms if external to the package. Fully qualify the subprogram with its package name when invoked externally from the package. Use the package_name.subprogram syntax.

Fully qualifying a subprogram when invoked within the same package is optional.

Example 1: Invokes the validate function from the reset_comm procedure within the same package. The package name prefix is not required; it is optional.

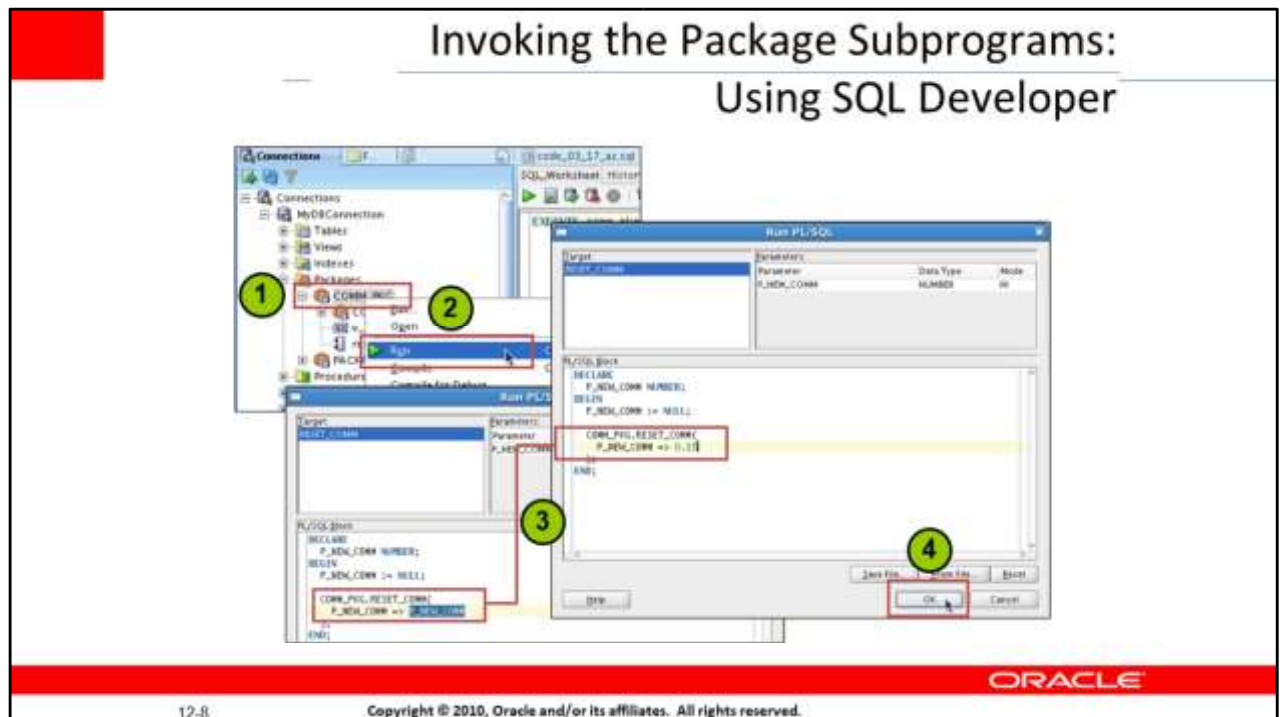
Example 2: Calls the reset_comm procedure from SQL*Plus (an environment external to the package) to reset the prevailing commission to 0.15 for the user session.

Example 3: Calls the reset_comm procedure that is owned by a schema user called SCOTT. Using SQL*Plus, the qualified package procedure is prefixed with the schema name. This can be simplified by using a synonym that references the schema.package_name.

Assume that a database link named NY has been created for a remote database

in which the reset_comm package procedure is created. To invoke the remote procedure, use:

```
EXECUTE comm_pkg.reset_comm@NY(0.15)
```



Invoking the Package Subprograms: Using SQL Developer

You can use SQL Developer to invoke a package's subprogram as follows: 1.

Right-click the package's name in the Packages node in the Navigation tree.

2. Select Run from the floating menu. The Run PL/SQL window is displayed. You can use the Run PL/SQL window to specify parameter values for running a PL/SQL function or procedure. (If you specify a package, select a function or procedure in the package.) Specify the following:

- Target: Select the name of the function or procedure to run.
- Parameters: This section lists each parameter for the specified target. The mode of each parameter can be IN (the value is passed in), OUT (the value is returned), or IN/OUT (the value is passed in, and the result of the function or procedure's action is stored in the parameter).

3. In the PL/SQL Block section, change the formal IN and IN/OUT parameter specifications in this block to actual values that you want to use for running the function or procedure. For example, to

Program Units 3 - 8

specify 0.15 as the value for an input parameter named

P_NEW_COMM, change P_NEW_COMM=>P_NEW_COMM to P_NEW_COMM=>0.15.

4. Click OK. SQL Developer runs the function or procedure.

Creating and Using Bodiless Packages

```
CREATE OR REPLACE PACKAGE global_consts IS
  c_mile_2_kilo    CONSTANT NUMBER := 1.6093;
  c_kilo_2_mile    CONSTANT NUMBER := 0.6214;
  c_yard_2_meter   CONSTANT NUMBER := 0.9144;
  c_meter_2_yard   CONSTANT NUMBER := 1.0936;
END global_consts;
```

```
SET SERVEROUTPUT ON
BEGIN
  DBMS_OUTPUT.PUT_LINE('20 miles = ' ||
    20 * global_consts.c_mile_2_kilo || ' km');
END;
```

```
SET SERVEROUTPUT ON
CREATE FUNCTION mtr2yrd(p_m NUMBER) RETURN NUMBER IS
BEGIN
  RETURN (p_m * global_consts.c_meter_2_yard);
END mtr2yrd;
/
EXECUTE DBMS_OUTPUT.PUT_LINE(mtr2yrd(1))
```

ORACLE

12-9

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

Creating and Using Bodiless Packages

The variables and constants declared within stand-alone subprograms exist only for the duration that the subprogram executes. To provide data that exists for the duration of the user session, create a package specification containing public (global) variables and constant declarations. In this case, create a package specification without a package body, known as a bodiless package. As discussed earlier in this lesson, if a specification declares only types, constants, variables, and exceptions, then the package body is unnecessary.

Examples

The first code box in the slide creates a bodiless package specification with several constants to be used for conversion rates. A package body is not required to support this package specification. It is assumed that the SET SERVEROUTPUT ON statement was issued before executing the code examples in the slide.

The second code box references the c_mile_2_kilo constant in the global_consts package by prefixing the package name to the identifier of the constant.

The third example creates a stand-alone function c_mtr2yrd to convert meters to yards, and uses the constant conversion rate c_meter_2_yard

Program Units 3 - 9

declared in the `global_consts` package. The function is invoked in a `DBMS_OUTPUT.PUT_LINE` parameter.

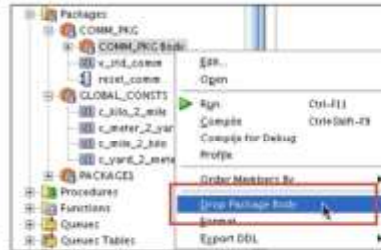
Rule to be followed: When referencing a variable, cursor, constant, or exception from outside the package, you must qualify it with the name of the package.

Removing Packages: Using SQL Developer or the SQL DROP Statement

Drop package specification and body



Drop package body only



```
-- Remove the package specification and body  
DROP PACKAGE package_name;
```

```
-- Remove the package body only  
DROP PACKAGE BODY package_name;
```

ORACLE

12-10

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

Removing Packages

When a package is no longer required, you can use a SQL statement in SQL Developer to remove it. A package has two parts; therefore, you can remove the whole package, or you can remove only the package body and retain the package specification.

Viewing Packages Using the Data Dictionary

```
-- View the package specification.
SELECT text
FROM   user_source
WHERE  name = 'COMM_PKG' AND type = 'PACKAGE'
ORDER BY LINE;
```


TEXT

```
1 PACKAGE comm_pkg IS
2   cmt_count NUMBER := 0; --initialized to 0
3   PROCEDURE reset_comm(new_comm NUMBER);
4 END comm_pkg;
```

```
-- View the package body.
SELECT text
FROM   user_source
WHERE  name = 'COMM_PKG' AND type = 'PACKAGE BODY'
ORDER BY LINE;
```

TEXT

```
1 PACKAGE BODY comm_pkg IS
2   FUNCTION validate_comm(NUMBER) RETURN BOOLEAN IS
3   max_comm employees.commission_pct/2;
4   BEGIN
5     SELECT MAX(commission_pct) INTO max_comm
6     FROM   employees;
7     RETURN (comm BETWEEN 0.0 AND max_comm);
```



12-11
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Viewing Packages in the Data Dictionary

The source code for PL/SQL packages is also stored in the USER_SOURCE and ALL_SOURCE data dictionary views. The USER_SOURCE table is used to display PL/SQL code that you own. The ALL_SOURCE table is used to display PL/SQL code to which you have been granted the EXECUTE right by the owner of that subprogram code and provides an OWNER column in addition to the preceding columns.

When querying the package, use a condition in which the TYPE column is: Equal to 'PACKAGE' to display the source code for the package specification

Equal to 'PACKAGE BODY' to display the source code for the package body

You can also view the package specification and body in SQL Developer using the package name in the Packages node.

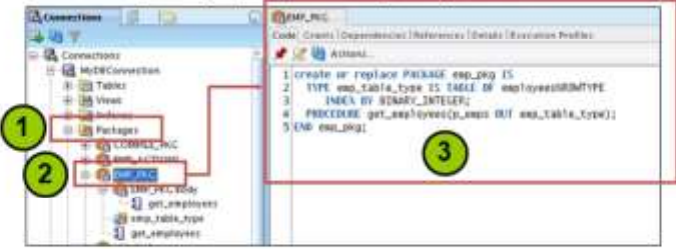
Note: You cannot display the source code for Oracle PL/SQL built-in packages, or PL/SQL whose source code has been wrapped by using a WRAP utility or obfuscation. Obfuscating and wrapping PL/SQL source code is covered in a later lesson. Clicking the Execute Statement (F9) icon (instead of the Run Script icon) in the SQL Worksheet toolbar, sometimes displays a

Program Units 3 - 11

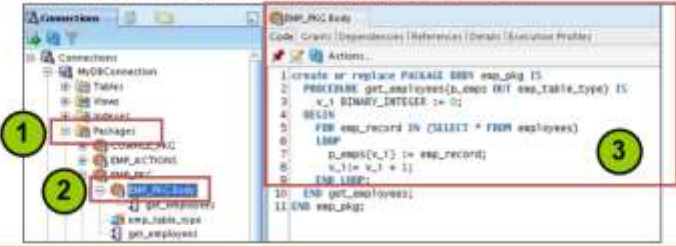
better formatted output in the Results tab as shown in the slide examples.

Viewing Packages Using SQL Developer

To view the package spec, click the package name



To view the package body, click the package body



ORACLE

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

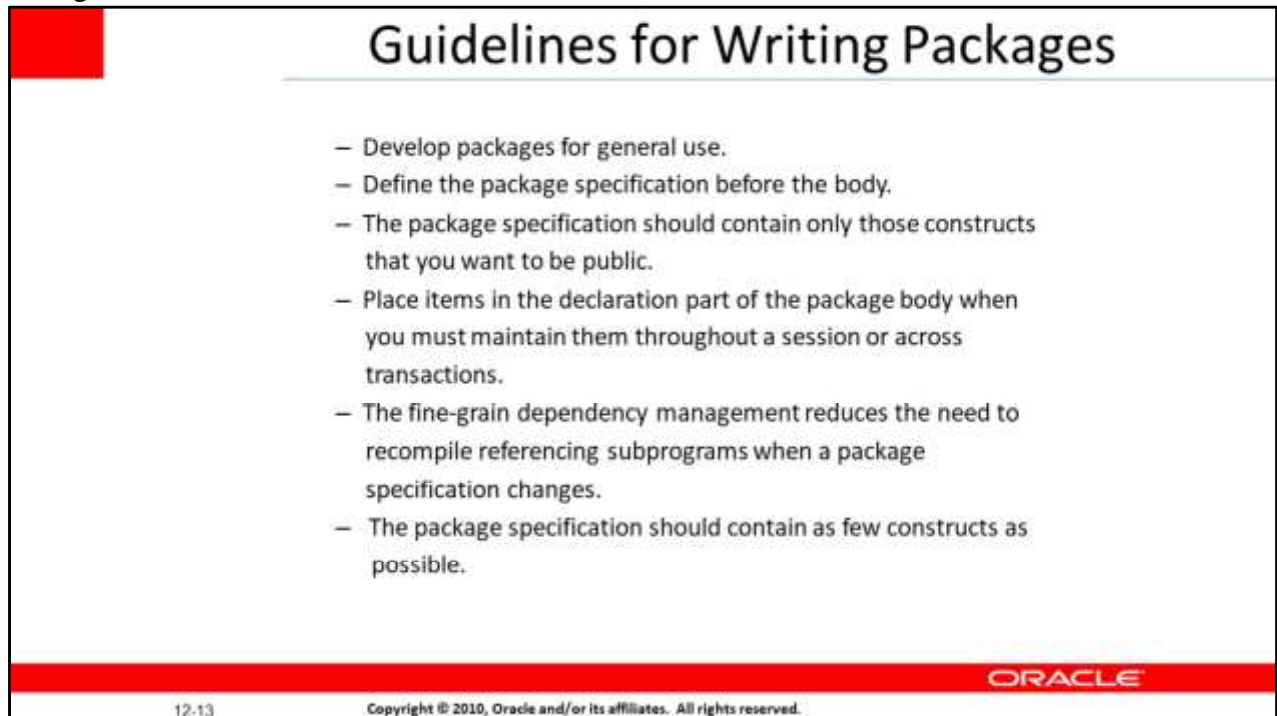
Viewing Packages Using SQL Developer

To view a package's spec in SQL Developer, use the following steps:

1. Click the Packages node in the Connections tab.
2. Click the package's name.
3. The package's spec code is displayed in the Code tab as shown in the slide.

To view a package's body in SQL Developer, use the following steps:

1. Click the Packages node in the Connections tab.
2. Click the package's body.
3. The package's body code is displayed in the Code tab as shown in the slide.



Guidelines for Writing Packages

- Develop 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.
- The fine-grain dependency management reduces the need to recompile referencing subprograms when a package specification changes.
- The package specification should contain as few constructs as possible.

ORACLE

12-13 Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Guidelines for Writing Packages

Keep your packages as general as possible, so that they can be reused in future applications. Also, avoid writing packages that duplicate features provided by the Oracle server.

Package specifications reflect the design of your application, so define them before defining the package bodies. The package specification should contain only those constructs that must be visible to the users of the package. Thus, other developers cannot misuse the package by basing code on irrelevant details.

Place items in the declaration part of the package body when you must maintain them throughout a session or across transactions. For example, declare a variable called `NUMBER_EMPLOYED` as a private variable if each call to a procedure that uses the variable needs to be maintained. When declared as a global variable in the package specification, the value of that global variable is initialized in a session the first time a construct from the package is invoked.

Before Oracle Database 11g, changes to the package body did not require recompilation of dependent constructs, whereas changes to the package specification required the recompilation of every stored subprogram that

Oracle Database: Develop PL/SQL

Program Units 3 - 13

references the package. Oracle Database 11g reduces this dependency. Dependencies are now tracked at the level of element within unit. Fine-Grain Dependency Management is covered in a later lesson.

