Creación de Paquetes

Objetivos

Al finalizar esta lección, debería estar capacitado para hacer lo siguiente:

- **Describir los paquetes y enumerar sus posibles componentes**
- **Crear un paquete para agrupar las variables, los cursores, las constantes, las excepciones, los procedimientos y las funciones relacionadas**
- **Designar la construcción de un paquete como pública o privada**
- **Llamar a una construcción de un paquete**
- **Describir un uso de un paquete sin cuerpo**

ORACLE

Visión General de los Paquetes

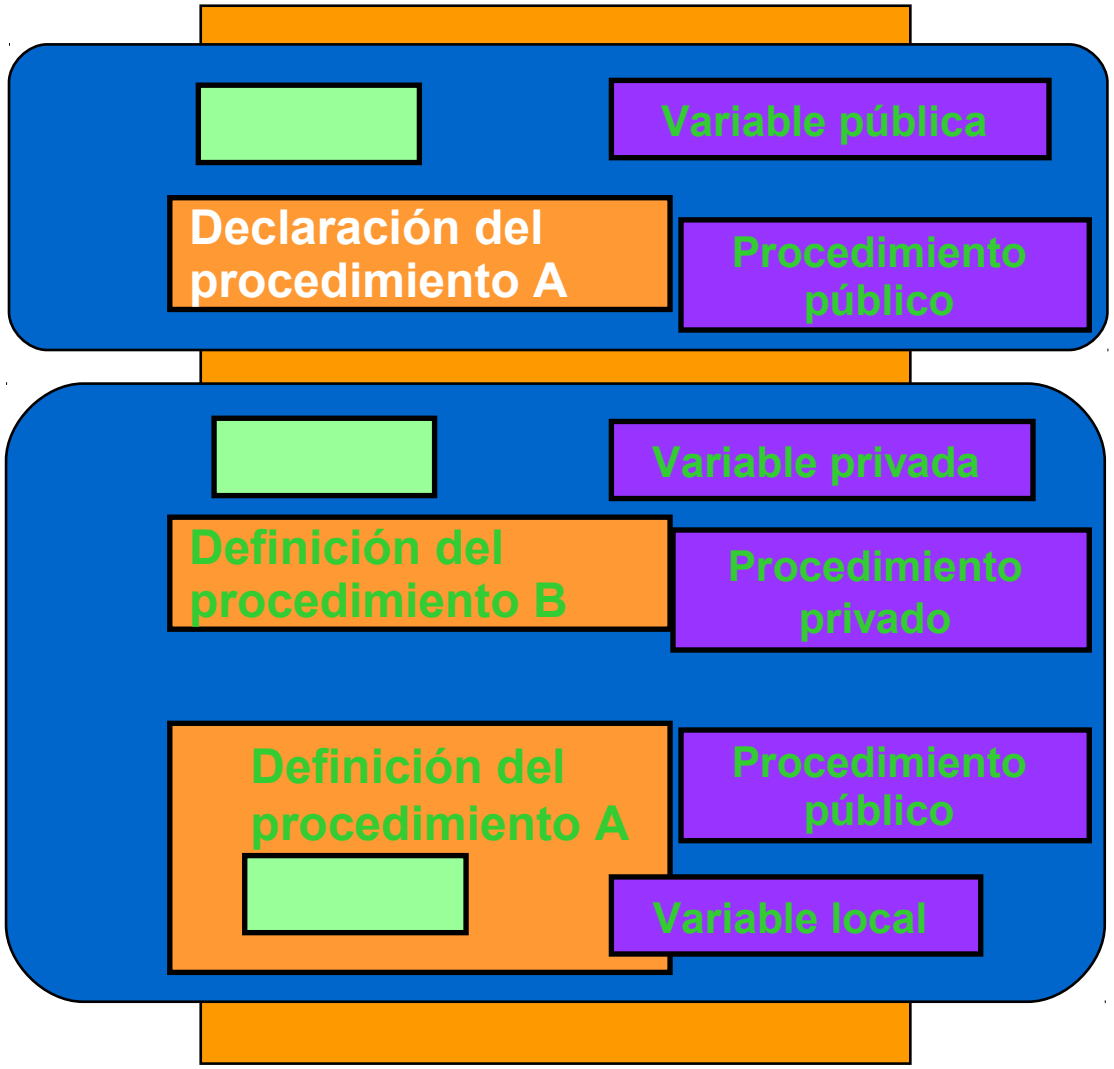
Paquetes:

- **Agrupaciones lógicas de tipos, elementos y subprogramas PL/SQL**
- **Constan de dos partes:**
  - **Especificación**
  - **Cuerpo**
- **No se pueden llamar, parametrizar ni anidar**
- **Permiten a Oracle Server leer varios objetos a la vez en la memoria**

Componentes de un Paquete

Especificación del paquete

Cuerpo del paquete



Componentes de un Paquete

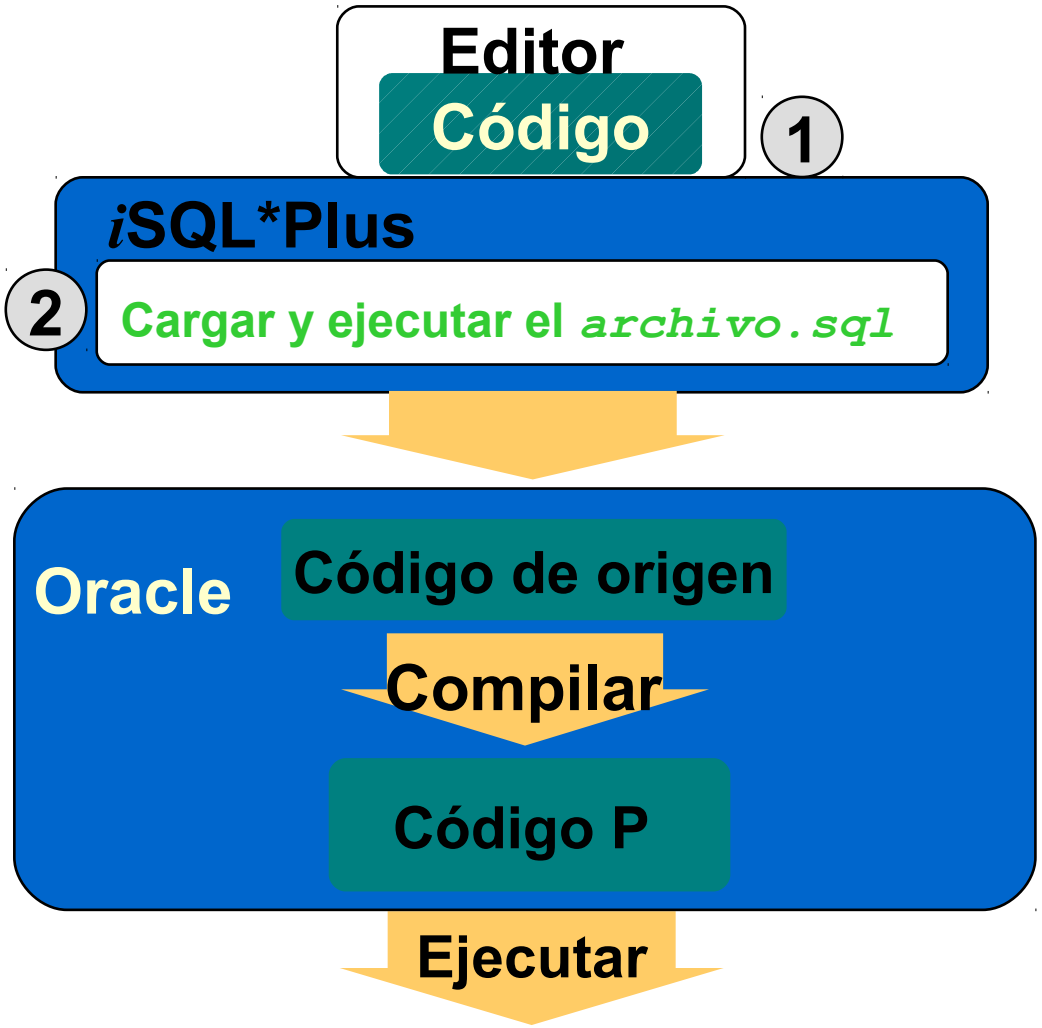
Las construcciones de paquetes públicos son las que se declaran en la especificación del paquete y se definen en el cuerpo del paquete. Las construcciones de paquetes privados son las que únicamente se definen en el cuerpo del paquete.

Ámbito de la Construcción	Descripción	Colocación en el Paquete
Pública	Se puede hacer referencia a ella desde cualquier entorno de Oracle Server	Declarada en la especificación del paquete y se puede definir en el cuerpo del paquete
Privada	Sólo pueden hacer referencia a ella otras construcciones que forman parte del mismo paquete	Declarada y definida en el cuerpo del paquete

Componentes de un Paquete

Visibilidad de la Construcción	Descripción
Local	Una variable que está definida en un subprograma que no es visible para los usuarios externos. Variable privada (local del paquete): Se pueden definir variables en el cuerpo de un paquete. A estas variables sólo pueden acceder otros objetos del mismo paquete. No son visibles para ningún subprograma ni objeto que esté fuera de la base de datos.
Global	Una variable o un subprograma al que se puede hacer referencia (y cambiar) en el exterior del paquete y que es visible para los usuarios externos. Los elementos globales de los paquetes se pueden declarar en la especificación del paquete.

Desarrollo de un Paquete



Desarrollo de un Paquete

- Si guarda el texto de la sentencia `CREATE PACKAGE` en dos archivos SQL diferentes, facilitará la modificación posterior del paquete.
- Puede haber una especificación de paquete sin un cuerpo de paquete, pero no un cuerpo de paquete sin una especificación de paquete.

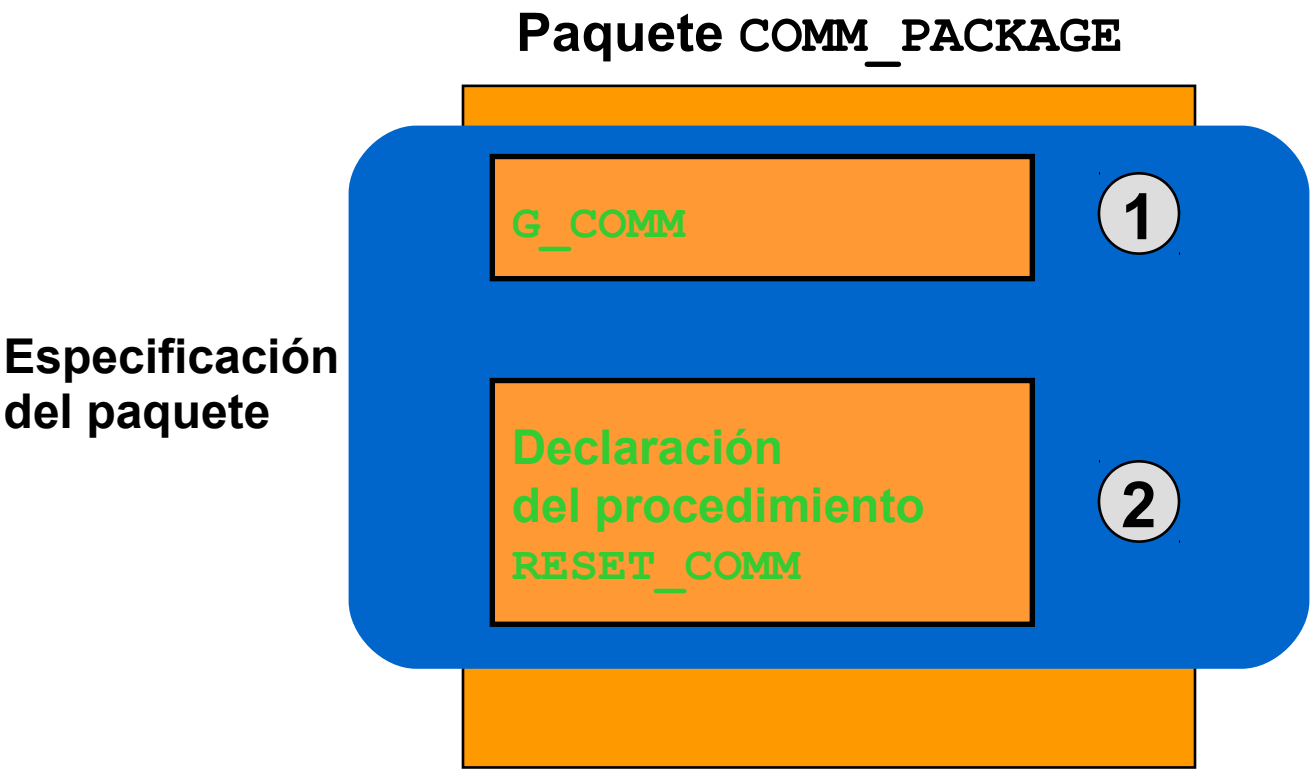
Creación de la Especificación del Paquete

Sintaxis:

```
CREATE [OR REPLACE] PACKAGE nombre_paquete
IS|AS
    declaraciones de los tipos y los elementos públicos
    especificaciones del subprograma
END nombre_paquete;
```

- La opción `REPLACE` borra y vuelve a crear la especificación del paquete.
- Las variables declaradas en la especificación del paquete se inicializan como `NULL` por defecto.
- Todas las construcciones declaradas en la especificación de un paquete son visibles para los usuarios que tienen privilegios sobre el paquete.

Declaración de Construcciones Públicas



G_COMM es una variable pública (global) y RESET_COMM es un procedimiento público.

Creación de la Especificación de un Paquete: Ejemplo

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

Package created.

- G_COMM es una variable global y se inicializa como 0.10.
- RESET_COMM es un procedimiento público que se implementa en el cuerpo del paquete.

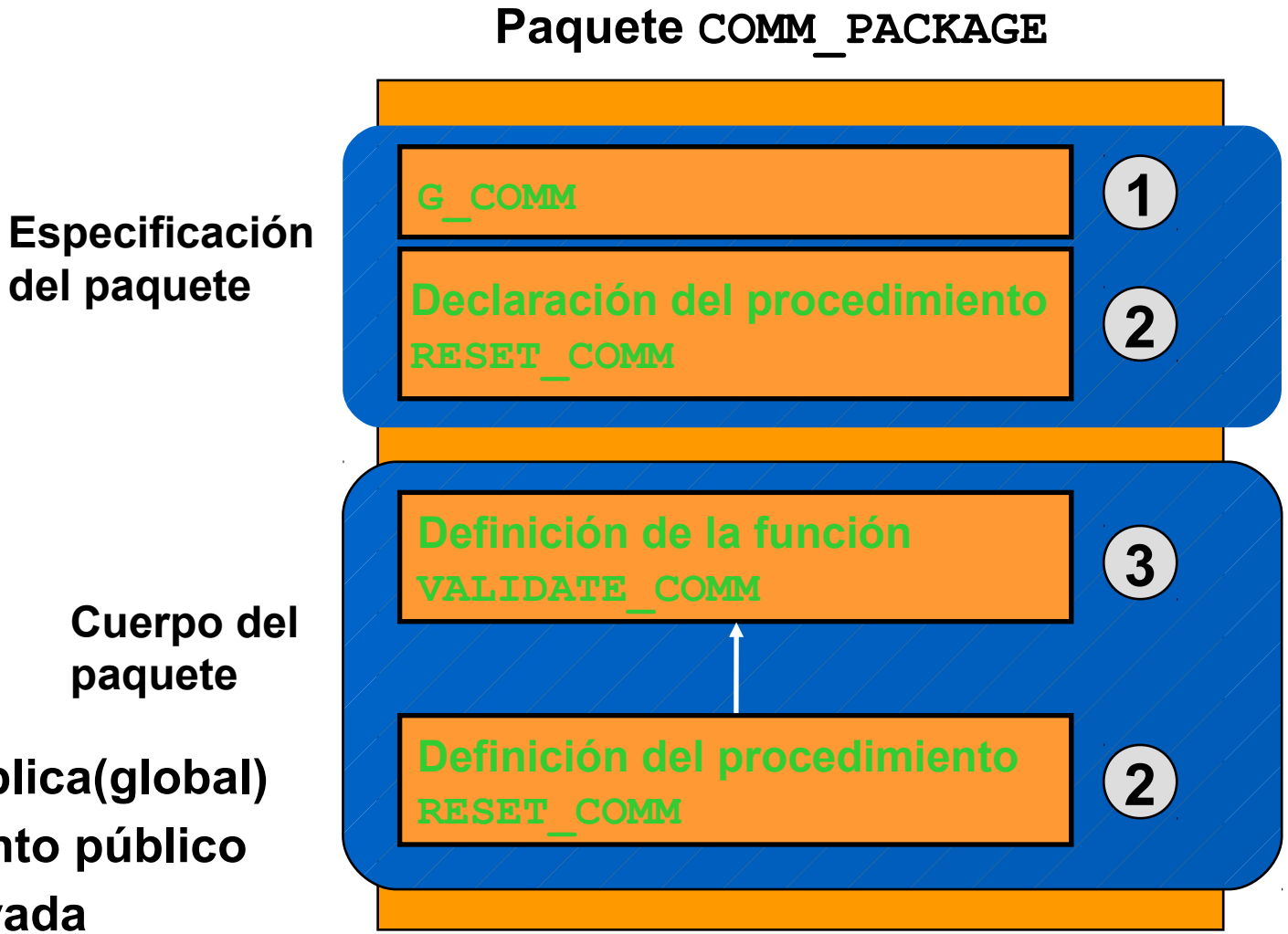
Creación del Cuerpo del Paquete

Sintaxis:

```
CREATE [OR REPLACE] PACKAGE BODY nombre_paquete
IS|AS
    declaraciones de los tipos y los elementos privados
    cuerpos de los subprogramas
END nombre_paquete;
```

- La opción `REPLACE` borra y vuelve a crear el cuerpo del paquete.
- Los identificadores que sólo están definidos en el cuerpo del paquete son construcciones privadas. No son visibles fuera del cuerpo de paquete.
- Todas las construcciones privadas se deben declarar antes de utilizarlas en las construcciones públicas.

Construcciones Públicas y Privadas



1 = variable pública(global)
2 = procedimiento público
3 = función privada

Creación del Cuerpo de un Paquete: Ejemplo

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;
...
```

Creación del Cuerpo de un Paquete: Ejemplo

comm_pack.sql

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

Package body created.

Llamadas a Construcciones de Paquetes

Ejemplo 1: Se llama a una función desde un procedimiento en el interior del mismo paquete.

```
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;
```

Llamadas a Construcciones de Paquetes

Ejemplo 2: Se llama a un procedimiento de paquete desde *i*SQL*Plus.

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

Ejemplo 3: Se llama a un procedimiento de paquete de un esquema diferente.

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

Ejemplo 4: Se llama a un procedimiento de paquete de una base de datos remota.

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

Instrucciones para Desarrollar Paquetes

- Construya paquetes de uso general.
- Defina la especificación del paquete antes que el cuerpo.
- La especificación del paquete sólo debería contener aquellas construcciones que desee que sean públicas.
- Coloque elementos en la sección de declaración del cuerpo del paquete cuando deba mantenerlos en toda una sesión o entre transacciones.
- Para realizar cambios en la especificación del paquete, es necesario volver a compilar cada subprograma que haga referencia a él.
- La especificación del paquete debería contener la menor cantidad de construcciones posible.

Uso de Declaraciones Posteriores

Hay que declarar los identificadores antes de hacer referencia a ellos.

```
CREATE OR REPLACE PACKAGE BODY forward_pack
IS
   PROCEDURE award_bonus(. . .)
   IS
   BEGIN
    calc_rating(. . .);        --referencia ilegal

   END;

   PROCEDURE calc_rating(. . .)
   IS
   BEGIN
      ...
   END;

END forward_pack;
/
```

Uso de Declaraciones Posteriores

```
CREATE OR REPLACE PACKAGE BODY forward_pack
IS

PROCEDURE calc_rating(. . .);     -- declaración posterior


PROCEDURE award_bonus(. . .)
IS                                -- subprogramas
definidos
BEGIN                             -- en orden alfabético
  calc_rating(. . .);
. . .
END;

PROCEDURE calc_rating(. . .)
IS
BEGIN
  . . .
END;

END forward_pack;
/
```

Creación de cursores en paquetes

- **El uso de cursores dentro del paquete es un poco especial**
- **En la cabecera se pone el nombre del cursor y el tipo de datos que devuelve. Por ejemplo,**

```
TYPE tcursor IS REF CURSOR
```

- **En el cuerpo, se completa la declaración. Por ejemplo,**

```
c_depart  tcursor ;
BEGIN
    OPEN c_depart FOR
       SELECT * FROM depart;
…
END;
```

Sobrecarga

- Permite utilizar el mismo nombre para diferentes subprogramas en el interior de un bloque PL/SQL, un subprograma o un paquete
- Es necesario que el número, el orden y la familia del tipo de dato de los parámetros formales de los subprogramas sean diferentes
- Permite obtener más flexibilidad porque ni el usuario ni la aplicación están limitados por el tipo de dato o el número específico de los parámetros formales.

Nota: Sólo se pueden sobrecargar los subprogramas empaquetados o locales. No se puede sobrecargar los subprogramas autónomos.

ORACLE

Sobrecarga: Ejemplo

over_pack.sql

```
CREATE OR REPLACE PACKAGE over_pack
IS
  PROCEDURE add_dept
    (p_deptno IN departments.department_id%TYPE,
     p_name IN departments.department_name%TYPE
                                    DEFAULT 'unknown',
     p_loc IN departments.location_id%TYPE DEFAULT 0);
  PROCEDURE add_dept
    (p_name IN departments.department_name%TYPE
                                    DEFAULT 'unknown',
     p_loc  IN departments.location_id%TYPE DEFAULT 0);
END over_pack;
/
```

Package created.

Sobrecarga: Ejemplo

over_pack_body.sql

```
CREATE OR REPLACE PACKAGE BODY over_pack  IS
  PROCEDURE add_dept
  (p_deptno IN departments.department_id%TYPE,
   p_name IN departments.department_name%TYPE DEFAULT 'unknown',
   p_loc  IN departments.location_id%TYPE DEFAULT 0)
  IS
  BEGIN
   INSERT INTO departments (department_id,
                            department_name, location_id)
   VALUES  (p_deptno, p_name, p_loc);
  END add_dept;
  PROCEDURE add_dept
  (p_name IN departments.department_name%TYPE DEFAULT 'unknown',
   p_loc  IN departments.location_id%TYPE DEFAULT 0)
  IS
  BEGIN
   INSERT INTO departments (department_id,
                            department_name, location_id)
   VALUES (departments_seq.NEXTVAL, p_name, p_loc);
  END add_dept;
END over_pack;
/
```

Sobrecarga: Ejemplo

- La mayoría de las funciones incorporadas están sobrecargadas.
- Por ejemplo, observe la función `TO_CHAR` del paquete `STANDARD`.

```
FUNCTION TO_CHAR (p1 DATE) RETURN VARCHAR2;
FUNCTION TO_CHAR (p2 NUMBER) RETURN VARCHAR2;
FUNCTION TO_CHAR (p1 DATE, P2 VARCHAR2) RETURN VARCHAR2;
FUNCTION TO_CHAR (p1 NUMBER, P2 VARCHAR2) RETURN VARCHAR2;
```

- Si se vuelve a declarar un subprograma incorporado en un programa PL/SQL, la declaración local sustituye a la declaración global.

Declaración de un Paquete sin Cuerpo

```
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.
```

Referencia a una Variable Pública desde un Procedimiento Autónomo

Ejemplo:

```
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

Eliminación de Paquetes

Para eliminar la especificación y el cuerpo del paquete, utilice la siguiente sintaxis:

```
DROP PACKAGE nombre_paquete;
```

Para eliminar el cuerpo del paquete, utilice la siguiente sintaxis:

```
DROP PACKAGE BODY nombre_paquete;
```

Ventajas de los Paquetes

- **Funcionalidad agregada: Persistencia de variables y cursores**
- **Mejor rendimiento:**
  - **El paquete completo se carga en memoria la primera vez que se hace referencia a él.**
  - **Sólo hay una copia en memoria para todos los usuarios.**
  - **La jerarquía de dependencia se simplifica.**
- **Sobrecarga: Varios subprogramas con el mismo nombre.**

Ventajas de los Paquetes

- **Modularidad: Encapsulan construcciones relacionadas.**
- **Facilidad del diseño de la aplicación: Codifican y compilan la especificación y el cuerpo por separado.**
- **Ocultación de información:**
  - **Sólo son visibles y accesibles a las aplicaciones las declaraciones de la especificación del paquete.**
  - **Las construcciones privadas del cuerpo del paquete están ocultas y son inaccesibles.**
  - **Todo el código está oculto en el cuerpo del paquete.**

Resumen

En esta lección, ha aprendido a:

- **Agrupar procedimientos y funciones relacionados en un paquete**
- **Cambiar el cuerpo de un paquete sin influir en la especificación del paquete**
- **Sobrecargar subprogramas**

Resumen

Comando	Tarea
`CREATE [OR REPLACE] PACKAGE`	Crear (o modificar) la especificación de un paquete existente
`CREATE [OR REPLACE] PACKAGE BODY`	Crear (o modificar) el cuerpo de un paquete existente
`DROP PACKAGE`	Eliminar la especificación del paquete y el cuerpo del paquete
`DROP PACKAGE BODY`	Eliminar sólo el cuerpo del paquete