12

Creación de Paquetes

ORACLE

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

Finalidad de la Lección

En esta lección aprenderá qué es un paquete y cuáles son sus componentes. También se le explicará cómo crear y utilizar los paquetes.

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**

ORACLE

Visión General de los Paquetes

Los paquetes agrupan tipos, elementos y subprogramas PL/SQL relacionados en un contenedor. Por ejemplo, un paquete de Recursos Humanos puede contener los procedimientos de contratación y de despido, las funciones para calcular las comisiones y las bonificaciones, y las variables de exención de impuestos.

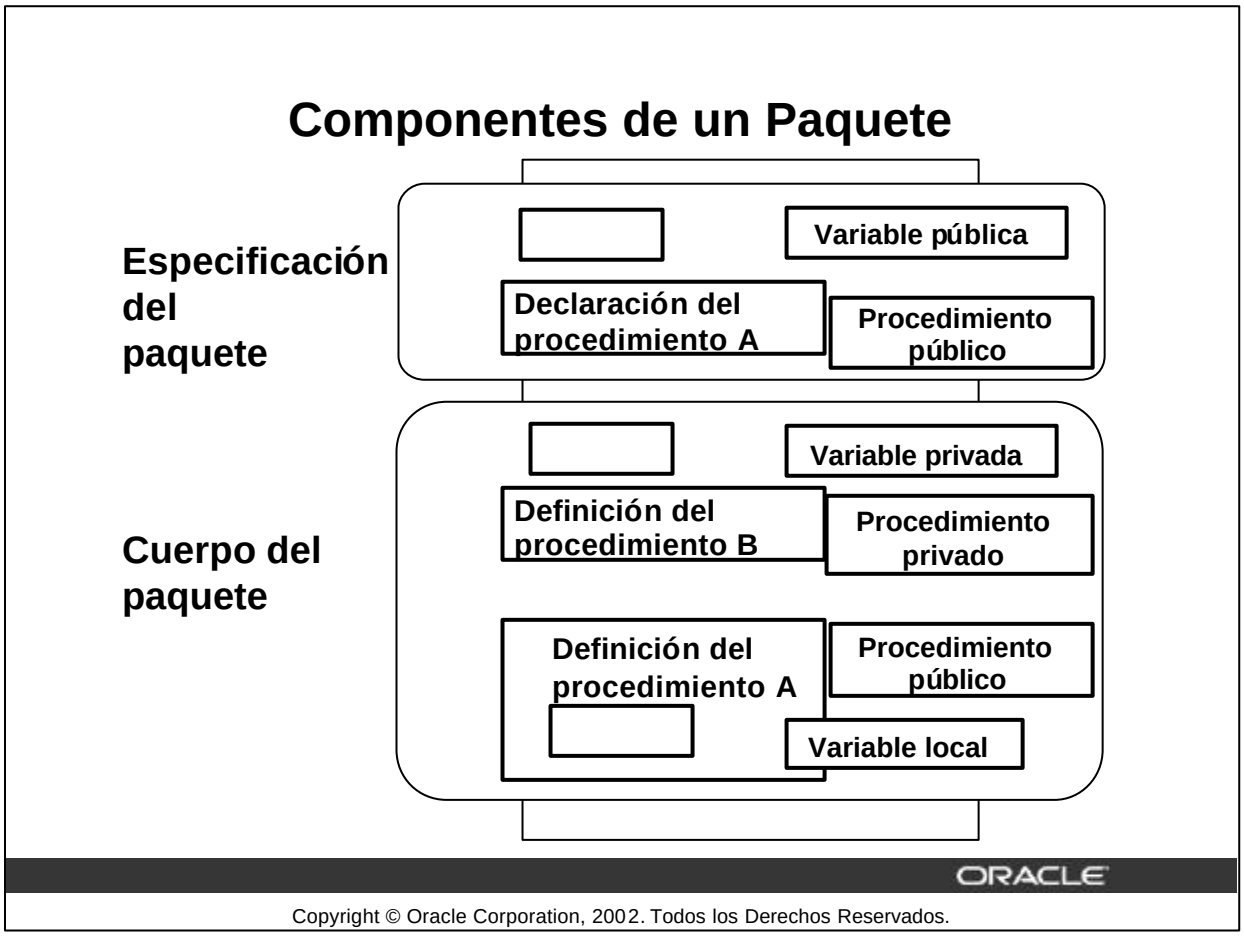
Los paquetes suelen tener una especificación y un cuerpo, que se almacenan por separado en la base de datos.

La especificación es la interfaz de las aplicaciones. Declara los tipos, las variables, las constantes, las excepciones, los cursores y los subprogramas que se pueden utilizar. La especificación del paquete puede incluir también PRAGMAs, que son directivas para el compilador.

El cuerpo define por completo los cursores y los subprogramas y, de esa manera, implementa la especificación.

Los paquetes no se pueden llamar, parametrizar ni anidar. Sin embargo, el formato de los paquetes es similar al de los subprogramas. Una vez escritos y compilados, muchas aplicaciones pueden compartir el contenido.

La primera vez que se llama a una construcción PL/SQL empaquetada, el paquete completo se carga en memoria. De esta manera, las posteriores llamadas a las construcciones de ese mismo paquete no requiere E/S (entrada/salida) del disco.



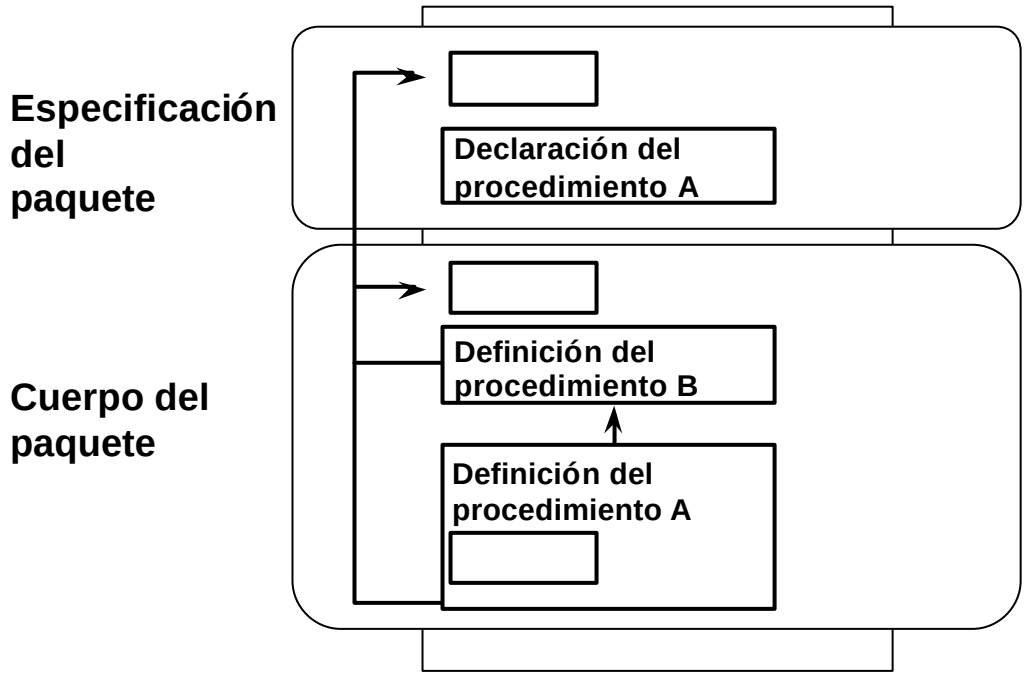
Desarrollo de Paquetes

El proceso de creación de un paquete consta de dos partes: primero la especificación del paquete y luego el cuerpo del 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

Nota: Oracle Server almacena la especificación y el cuerpo de un paquete en la base de datos por separado. De esta manera, se puede cambiar la definición de una construcción de programa en el cuerpo del paquete sin que Oracle Server invalide otros objetos de esquema que llaman o hacen referencia a esa construcción de programa.

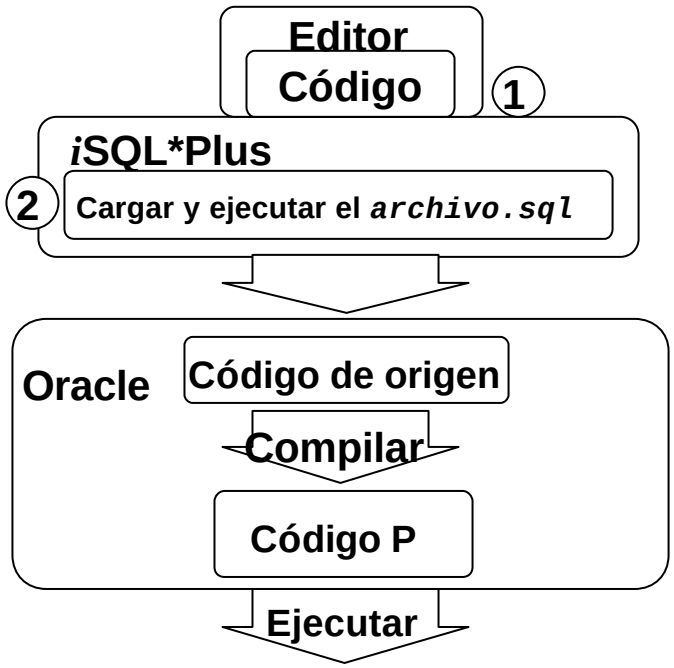
Referencia a Objetos de Paquetes



Desarrollo de Paquetes (continuación)

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



ORACLE

Cómo Desarrollar un Paquete

1. Escriba la sintaxis: Introduzca el código en un editor de texto y guárdelo como un archivo de comandos SQL.
2. Compile el código: Ejecute el archivo de comandos SQL para generar y compilar el código de origen. El código de origen se compila en el código P.

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.**

ORACLE

Cómo Desarrollar un Paquete

Existen tres pasos básicos para desarrollar un paquete, que son similares a los que se utilizan para desarrollar un procedimiento autónomo.

1. Escriba el texto de la sentencia CREATE PACKAGE en un archivo de comandos SQL para crear la especificación del paquete y ejecute el archivo de comandos. El código de origen se compila en el código P y se almacena en el diccionario de datos.
2. Escriba el texto de la sentencia CREATE PACKAGE BODY en un archivo de comandos SQL para crear el cuerpo del paquete y ejecute el archivo de comandos.
   El código de origen se compila en el código P y también se almacena en el diccionario de datos.
3. Llame a cualquier construcción pública del paquete desde un entorno de Oracle Server.

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.**

ORACLE

Cómo Crear la Especificación de un Paquete

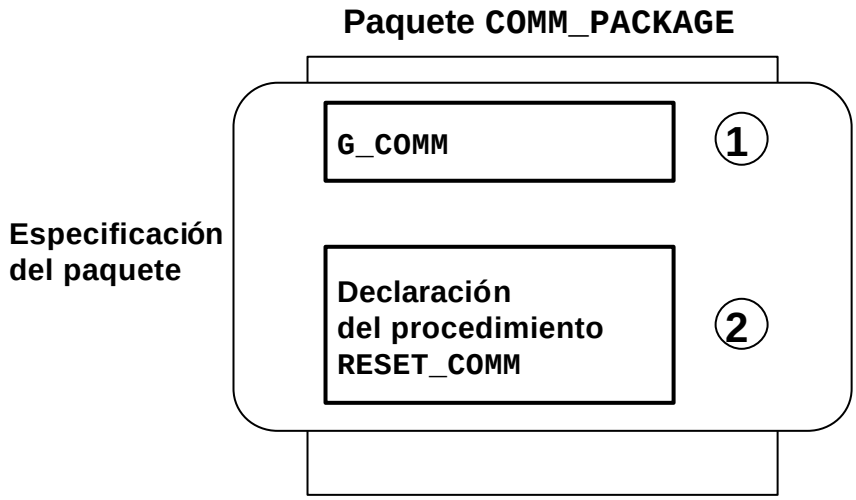
Para crear paquetes, hay que declarar todas las construcciones públicas en el interior de la especificación del paquete.

- Especifique la opción REPLACE si la especificación del paquete ya existe.
- Inicialice una variable con un valor constante o una fórmula en la declaración, si es necesario. De lo contrario, la variable se inicializa implícitamente como NULL.

Definición de la Sintaxis

Parámetro	Descripción
nombre_paquete	Nombre del paquete
declaraciones de los tipos y los	Declara variables, constantes, cursores, excepciones o tipos
especificaciones del subprograma	Declara los subprogramas PL/SQL

Declaración de Construcciones Públicas



ORACLE

Ejemplo de la Especificación de un Paquete

En la transparencia, `G_COMM` es una variable pública (global) y `RESET_COMM` es un procedimiento público.

En la especificación del paquete se declaran las variables públicas, los procedimientos públicos y las funciones públicas.

Los procedimientos o las funciones públicas son rutinas que otras construcciones del mismo paquete o del exterior del paquete pueden llamar de forma repetida.

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.**

ORACLE

Especificación del Paquete COMM_PACKAGE

En la transparencia, la variable G_COMM y el procedimiento RESET_COMM son construcciones públicas.

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.**

ORACLE

Creación del Cuerpo del Paquete

Para crear paquetes, hay que definir todas las construcciones públicas y privadas en el interior del cuerpo del paquete.

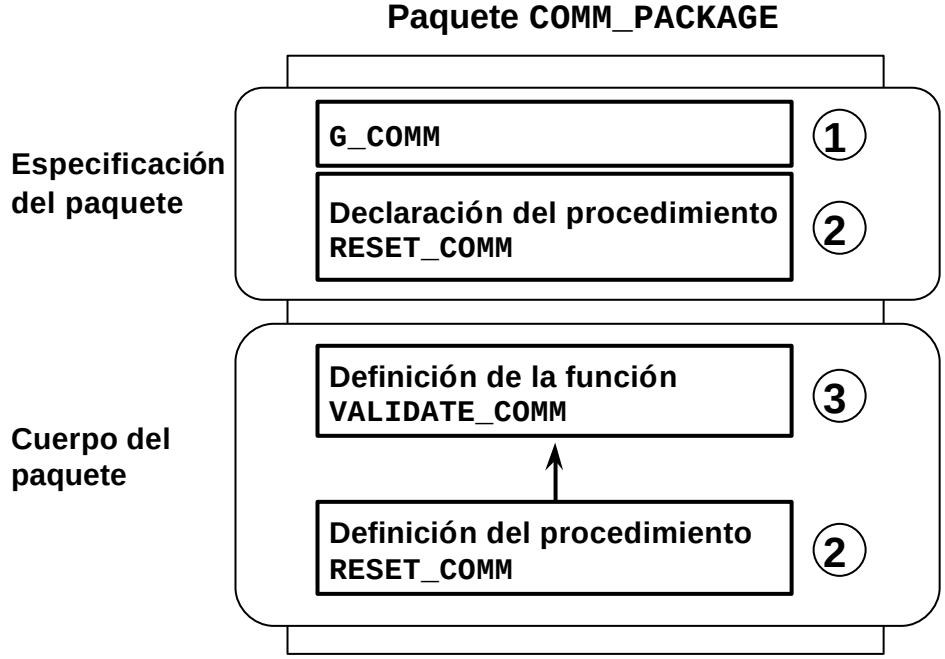
- Especifique la opción REPLACE si el cuerpo del paquete ya existe.
- El orden en el que se definen los subprogramas en el cuerpo del paquete es importante: debe declarar una variable antes que otra variable o subprograma se refiera a ella, y debe declarar o definir subprogramas privados antes de llamarlos desde otros subprogramas. En el cuerpo del paquete es habitual ver todas las variables y los subprogramas definidos en primer lugar y los subprogramas públicos en último lugar.

Definición de la Sintaxis

Defina todos los procedimientos y las funciones públicas y privadas en el cuerpo del paquete.

Parámetro	Descripción
nombre_paquete	Es el nombre del paquete
declaraciones de los tipos y los	Declara variables, constantes, cursores, excepciones o tipos
cuerpos de los subprogramas	Define subprogramas PL/SQL, públicos y privados

Construcciones Públicas y Privadas



Ejemplo de la Creación del Cuerpo de un Paquete

En la transparencia de esta página:

- 1 es una variable pública (global)
- 2 es un procedimiento público
- 3 es una función privada

Puede definir un procedimiento o una función privada para modularizar y clarificar el código de los procedimientos y las funciones públicas.

Nota: En la transparencia, la función privada aparece antes del procedimiento público. Al codificar el cuerpo del paquete, la definición de la función privada tiene que estar antes de la definición del procedimiento público.

Sólo las declaraciones de los subprogramas y los cursores sin cuerpo de una especificación de paquete poseen una implementación subyacente en el cuerpo del paquete. Por lo tanto, si una especificación sólo declara tipos, constantes, excepciones y especificaciones de llamadas, el cuerpo del paquete es innecesario. Sin embargo, se puede utilizar para inicializar los elementos declarados en la especificación del paquete.

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;
...
```

ORACLE

Cuerpo del Paquete para COMM_PACKAGE

Defina una función para validar la comisión. La comisión no puede ser mayor que la comisión más alta de todos los empleados existentes.

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.
```

ORACLE

Cuerpo del Paquete para COMM_PACKAGE (continuación)

Defina un procedimiento que le permita restablecer y validar la comisión predominante.

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;
```

ORACLE

Llamadas a Construcciones de Paquetes

Después de almacenar el paquete en la base de datos, puede llamar a la construcción del paquete desde el interior o el exterior del paquete, dependiendo de si la construcción es pública o privada.

Cuando se llama a un procedimiento o una función de paquete desde el interior del mismo paquete, no hay que cualificar su nombre.

Ejemplo 1

Se llama a la función `VALIDATE_COMM` desde el procedimiento `RESET_COMM`. Los dos subprogramas se encuentran en el paquete `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)
```

ORACLE

Llamadas a Construcciones de Paquetes (continuación)

Cuando llame a un procedimiento o una función de paquete desde el exterior del paquete, debe cualificar su nombre con el nombre del paquete.

Ejemplo 2

Se llama al procedimiento `RESET_COMM` desde *i*SQL*Plus haciendo que la comisión predominante sea 0.15 para la sesión de usuario.

Ejemplo 3

Se llama al procedimiento `RESET_COMM`, ubicado en el esquema `SCOTT`, desde *i*SQL*Plus haciendo que la comisión predominante sea 0.15 para la sesión de usuario.

Ejemplo 4

Se llama al procedimiento `RESET_COMM`, ubicado en una base de datos remota que está determinada por el enlace de base de datos denominado `NY`, desde *i*SQL*Plus haciendo que la comisión predominante sea 0.15 para la sesión de usuario.

Utilice las reglas de nomenclatura habituales para llamar a un procedimiento que se encuentra en un esquema diferente, o en una base de datos diferente en otro nodo.

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.
```

ORACLE

Declaración de un Paquete sin Cuerpo

Puede declarar variables públicas (globales) que existan durante la sesión de usuario. Puede crear una especificación de paquete que no necesite un cuerpo de paquete. Como ya se ha explicado previamente en esta lección, si en una especificación sólo se declaran tipos, constantes, excepciones y especificaciones de llamadas, el cuerpo del paquete es innecesario.

Ejemplo

En el ejemplo de la transparencia, se define la especificación de un paquete que contiene varias tasas de conversión. Todos los identificadores globales se declaran como constantes.

No es necesario utilizar un cuerpo de paquete para apoyar esta especificación del paquete, ya que los detalles de implementación no son necesarios para ninguna de las construcciones de la especificación del paquete.

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

ORACLE

Ejemplo

Utilice el procedimiento METER_TO_YARD para convertir los metros en yardas utilizando la tasa de conversión empaquetada en GLOBAL_CONSTS.

Cuando hace referencia a una variable, un cursor, una constante o una excepción desde el exterior del paquete, debe cualificar su nombre con el nombre del paquete.

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;
```

ORACLE

Eliminación de un Paquete

Cuando un paquete ya no es necesario, puede utilizar una sentencia SQL en *i*SQL*Plus para borrarlo. Los paquetes tienen dos partes, por lo que puede borrar todo el paquete o sólo el cuerpo y conservar la especificación.

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.**

ORACLE

Instrucciones para Escribir Paquetes

Cree paquetes que sean lo más generales posibles para que se puedan reutilizar en aplicaciones futuras. Asimismo, no escriba paquetes que dupliquen las funciones que proporciona Oracle Server.

Las especificaciones del paquete reflejan el diseño de la aplicación. Por lo tanto, defina las especificaciones antes de definir los cuerpos de los paquetes.

La especificación del paquete debería contener sólo aquellas construcciones que deban ser visibles para otros usuarios del paquete. De esta manera, otros desarrolladores no pueden dar mal uso al paquete basando el código en detalles irrelevantes.

Coloque elementos en la sección de declaración del cuerpo del paquete cuando deba mantenerlos en toda una sesión o entre transacciones. Por ejemplo, declare una variable llamada `NUMBER_EMPLOYED` como variable privada, si se necesita mantener cada llamada a un procedimiento que utiliza la variable. Cuando se declara como variable global en la especificación del paquete, el valor de esa variable global se inicializa en una sesión la primera vez que se llama a una construcción desde el paquete.

Los cambios realizados en el cuerpo del paquete no requieren la recompilación de las construcciones dependientes, mientras que los cambios realizados en la especificación del paquete requieren la recompilación de todos los subprogramas almacenados que hagan referencia al paquete. Para reducir la necesidad de recompilación cuando se cambia el código, coloque la menor cantidad de construcciones posible en la especificación de un paquete.

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.**

ORACLE

Ventajas del Uso de Paquetes

Los paquetes son una alternativa a la creación de procedimientos y funciones como objetos de esquema autónomos y poseen numerosas ventajas.

Modularidad

Debe encapsular las estructuras de programación relacionadas lógicamente en un módulo con nombre. Los paquetes son fáciles de entender y la interfaz entre los paquetes es sencilla, clara y está bien definida.

Facilidad del Diseño de Aplicaciones

Todo lo que se necesita al principio es la información de la interfaz en la especificación del paquete. Puede codificar y compilar una especificación sin su cuerpo. A continuación, los subprogramas almacenados que hagan referencia al paquete también se pueden compilar. No es necesario que defina por completo el cuerpo del paquete hasta que no esté preparado para completar la aplicación.

Ocultación de Información

Puede decidir qué construcciones son públicas (visibles y accesibles) o privadas (ocultas e inaccesibles). Sólo las declaraciones de la especificación del paquete son visibles y accesibles a las aplicaciones. El cuerpo del paquete oculta la definición de las construcciones privadas, de manera que, si la definición cambia, este cambio sólo afecta al paquete (no a la aplicación ni a los programas de llamada). De esta manera, puede cambiar la implementación sin tener que volver a compilar los programas de llamada. Además, al ocultar la información de implementación a los usuarios, protege la integridad del 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.**

ORACLE

Ventajas del Uso de Paquetes (continuación)

Funcionalidad Agregada

Las variables y los cursores públicos empaquetados se mantienen durante toda una sesión. Por lo tanto, se pueden compartir entre todos los subprogramas que se ejecutan en el entorno. También permiten mantener datos a través de transacciones sin que sea necesario almacenarlos en la base de datos. Las construcciones privadas también se mantienen durante toda una sesión, pero sólo se puede acceder a ellas desde el paquete.

Mejor Rendimiento

La primera vez que se llama a un subprograma empaquetado, el paquete completo se carga en memoria. De esta manera, las posteriores llamadas a los subprogramas relacionados no necesitan más E/S del disco. Además, los subprogramas empaquetados no crean dependencias en cascada y, por lo tanto, evitan operaciones de compilación innecesarias.

Sobrecarga

Los paquetes permiten sobrecargar los procedimientos y las funciones, lo que quiere decir que se pueden crear varios subprogramas con el mismo nombre en el mismo paquete, y cada uno puede aceptar parámetros con un número o un tipo de dato diferente.

Resumen

En esta lección, ha aprendido a:

- **Mejorar la organización, la gestión, la seguridad y el rendimiento utilizando paquetes**
- **Agrupar procedimientos y funciones relacionados en un paquete**
- **Cambiar el cuerpo de un paquete sin influir en la especificación del paquete**
- **Otorgar acceso de seguridad a todo el paquete**

ORACLE

Resumen

Los procedimientos y las funciones relacionadas se agrupan en paquetes. Los paquetes mejoran la organización, la gestión, la seguridad y el rendimiento.

Los paquetes se componen de una especificación y de un cuerpo. Puede cambiar el cuerpo de un paquete sin influir en su especificación.

Resumen

En esta lección, ha aprendido a:

- **Ocultar el código de origen a los usuarios**
- **Cargar un paquete completo en memoria en la primera llamada**
- **Reducir el acceso al disco en llamadas posteriores**
- **Proporcionar identificadores para la sesión de usuario**

ORACLE

Resumen (continuación)

Los paquetes permiten ocultar el código a los usuarios. La primera vez que se llama a un paquete, éste se carga por completo en memoria. De esta manera, se reduce el acceso al disco en las llamadas posteriores.

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

ORACLE

Resumen (continuación)

Puede crear, suprimir y modificar paquetes. Para eliminar la especificación y el cuerpo del paquete, se utiliza el comando DROP PACKAGE. Puede borrar el cuerpo de un paquete sin influir en su especificación.

Visión General de la Práctica 12

Esta práctica cubre los siguientes temas:

- **Creación de paquetes**
- **Llamadas a unidades de programa de paquetes**

ORACLE

Visión General de la Práctica 12

En esta práctica deberá crear especificaciones y cuerpos de paquetes. También tendrá que llamar a las construcciones que se encuentran en los paquetes utilizando datos de ejemplo.

Práctica 12

1. Cree la especificación y el cuerpo de un paquete denominado JOB_PACK. (Puede guardar el cuerpo y la especificación de dicho paquete en dos archivos diferentes.) Este paquete contiene los procedimientos ADD_JOB, UPD_JOB y DEL_JOB, así como la función Q_JOB.

   Nota: Para crear el paquete, utilice el código de los archivos de comandos que ha guardado anteriormente.

   a. Haga que todas las construcciones sean públicas.

      Note: Considere si sigue necesitando los procedimientos y las funciones autónomas que ha empaquetado.

   b. Llame al procedimiento ADD_JOB transfiriendo los valores IT_SYSAN y SYSTEMS ANALYST como parámetros.

   c. Consulte la tabla JOBS para ver los resultados.

JOB_ID	JOB_TITLE	MIN_SALARY	MAX_SALARY
IT_SYSAN	Systems Analyst		

2. Cree y llame a un paquete que contiene construcciones públicas y privadas.

   a. Cree la especificación y el cuerpo de un paquete denominado EMP_PACK que contiene el procedimiento NEW_EMP como construcción pública y la función VALID_DEPTID como construcción privada. (Puede guardar el cuerpo y la especificación en dos archivos diferentes.)

   b. Llame al procedimiento NEW_EMP utilizando el número de departamento 15. Dado que este número de departamento no existe en la tabla DEPARTMENTS, debería recibir un mensaje de error, tal y como se especifica en el manejador de excepciones del procedimiento.

   c. Llame al procedimiento NEW_EMP utilizando el identificador de departamento 80 existente.

Si tiene tiempo:

3. a. Cree un paquete llamado CHK_PACK que contenga los procedimientos CHK_HIREDATE y CHK_DEPT_MGR. Haga públicas las dos construcciones. (Puede guardar el cuerpo y la especificación en dos archivos diferentes.) El procedimiento CHK_HIREDATE comprueba si la fecha de contratación de un empleado se encuentra en el siguiente rango: [SYSDATE - 50 años, SYSDATE + 3 meses].

   Nota:

   - Si la fecha no es válida, debería emitir un error de aplicación con el mensaje adecuado, que indique por qué el valor no es aceptable.
   - Asegúrese de que se ignora el componente de tiempo en el valor de la fecha.
   - Utilice una constante para hacer referencia al límite de 50 años.
   - Los valores nulos en la fecha de contratación se deberían tratar como valores no válidos.

Práctica 12 (continuación)

El procedimiento CHK_DEPT_MGR comprueba la combinación de departamento y jefe de un empleado determinado. Este procedimiento CHK_DEPT_MGR acepta un identificador de empleado y un identificador de jefe. Además, comprueba si el jefe y el empleado trabajan en el mismo departamento. El procedimiento también comprueba si el puesto de trabajo del identificador de jefe es MANAGER.

Nota: Si la combinación del identificador de departamento y de jefe no es válida, debería emitir un error de aplicación con el mensaje adecuado

b. Pruebe el procedimiento CHK_HIREDATE con el siguiente comando:

```
EXECUTE chk_pack.chk_hiredate('01-JAN-47')
```

¿Qué ocurre y por qué?

c. Pruebe el procedimiento CHK_HIREDATE con el siguiente comando:

```
EXECUTE chk_pack.chk_hiredate(NULL)
```

¿Qué ocurre y por qué?

d. Pruebe el procedimiento CHK_DEPT_MGR con el siguiente comando:

```
EXECUTE chk_pack.chk_dept_mgr(117,100)
```

¿Qué ocurre y por qué?

13

Otros Conceptos sobre Paquetes

ORACLE

Objetivos

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

- **Escribir paquetes que utilizan la función de sobrecarga**
- **Describir errores con subprogramas que se hacen referencia mutuamente**
- **Inicializar variables con un procedimiento de un solo uso**
- **Identificar los estados persistentes**

ORACLE

Finalidad de la Lección

Esta lección explica funciones más avanzadas de PL/SQL como, por ejemplo, la sobrecarga, las referencias posteriores, los procedimientos de un solo uso y la persistencia de las variables, las constantes, las excepciones y los cursores. También examina el efecto de las funciones empaquetadas que se utilizan en las sentencias SQL.

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

Esta función permite definir diferentes subprogramas con el mismo nombre. Los subprogramas se distinguen por su nombre y sus parámetros. En ocasiones, el procesamiento de dos subprogramas es el mismo, pero los parámetros que se les transfieren son diferentes. En tal caso, es lógico que tengan el mismo nombre. PL/SQL comprueba los parámetros formales de los subprogramas para determinar a cuál de ellos se llama. Sólo se pueden sobrecargar los subprogramas empaquetados o locales. No se puede sobrecargar los subprogramas autónomos.

Restricciones

No puede sobrecargar:

- Dos subprogramas cuyos parámetros formales sólo se diferencien en el tipo de dato y estos tipos de dato pertenezcan a la misma familia (`NUMBER` y `DECIMAL`, por ejemplo, pertenecen a la misma familia).
- Dos subprogramas cuyos parámetros formales sólo se diferencien en el subtipo y estos subtipos estén basados en tipos de la misma familia (`VARCHAR` y `STRING`, por ejemplo, son subtipos PL/SQL de `VARCHAR2`)
- Dos funciones que sólo se diferencien en el tipo de retorno, incluso si los tipos pertenecen a diferentes familias.

Si sobrecarga subprogramas con las características anteriores, obtendrá un error en tiempo de ejecución.

Nota: Estas restricciones también son aplicables si los nombres de los parámetros también son iguales. Si los parámetros tienen nombres diferentes, puede llamar a los subprogramas utilizando la notación de nombres de los parámetros.

Sobrecarga (continuación)

Resolución de Llamadas

El compilador intenta encontrar una declaración que coincida con la llamada. Primero busca en el ámbito actual y luego, si es necesario, en los ámbitos delimitadores sucesivos. El compilador detiene la búsqueda si encuentra una o varias declaraciones de subprograma donde el nombre coincida con el nombre del subprograma llamado. En el caso de los subprogramas que tienen el mismo nombre y que se encuentran en el mismo nivel del ámbito, el compilador tiene que encontrar una coincidencia exacta en el número, el orden y el tipo de dato entre los parámetros reales y los formales.

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.
```

ORACLE

Sobrecarga: Ejemplo

La transparencia muestra la especificación de un paquete con procedimientos sobrecargados.

En el paquete se encuentra ADD_DEPT, que es el nombre de dos procedimientos sobrecargados. La primera definición acepta tres parámetros para poder insertar un nuevo departamento en la tabla de departamentos. La segunda definición sólo acepta dos parámetros, porque el identificador de departamento se rellena a lo largo de una secuencia.

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;
/
```

ORACLE

Ejemplo de Sobrecarga (continuación)

Si llama a ADD_DEPT proporcionando explícitamente un identificador de departamento, PL/SQL utiliza la primera versión del procedimiento. Si llama a ADD_DEPT sin ningún identificador de departamento, PL/SQL utiliza la segunda versión.

```
EXECUTE over_pack.add_dept (980,'Education',2500)
EXECUTE over_pack.add_dept ('Training', 2400)
SELECT * FROM departments
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
980	Education		2500

```
SELECT * FROM departments
WHERE department_name = 'Training';
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
320	Training		2400

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.**

ORACLE

Ejemplo de Sobrecarga (continuación)

La mayoría de las funciones incorporadas están sobrecargadas. Por ejemplo, la función TO_CHAR del paquete STANDARD posee cuatro declaraciones diferentes, tal y como se muestra en la transparencia. La función puede aceptar el tipo de dato DATE o NUMBER y convertirlo en un tipo de dato de carácter. El formato al que hay que convertir la fecha o el número también se puede especificar en la llamada de función.

Si se vuelve a declarar un subprograma incorporado en otro programa PL/SQL, la declaración local sustituye al subprograma estándar o incorporado. Para poder acceder al subprograma incorporado, necesita cualificarlo con el nombre del correspondiente paquete. Por ejemplo, si vuelve a declarar la función TO_CHAR, para acceder a la función incorporada, tendrá que hacer referencia a ella de la siguiente manera: STANDARD.TO_CHAR.

Si vuelve a declarar un subprograma incorporado como un subprograma autónomo, para poder acceder al subprograma, tendrá que cualificarlo con el nombre de su esquema como, por ejemplo, SCOTT.TO_CHAR.

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;
/
```

ORACLE

Uso de Declaraciones Posteriores

PL/SQL no permite el uso de referencias posteriores. Antes de utilizarlas, debe declarar los identificadores. Por lo tanto, hay que declarar un subprograma antes de llamarlo.

En el ejemplo de la transparencia, no se puede hacer referencia al procedimiento `CALC_RATING` porque todavía no se ha declarado. Para resolver el problema de la referencia ilegal, puede invertir el orden de los dos procedimientos. Sin embargo, esta solución tan fácil no siempre funciona. Suponga que los procedimientos se llaman entre sí o que no le queda más remedio que definirlos en orden alfabético.

PL/SQL permite utilizar una declaración de subprograma especial que se denomina declaración posterior. Esta declaración incluye la especificación del subprograma con un punto y coma (;) al final. Este tipo de declaraciones sirve para hacer lo siguiente:

- Definir subprogramas en orden lógico o alfabético
- Definir subprogramas mutuamente recursivos
- Agrupar subprogramas en un paquete

Los programas mutuamente recursivos son aquellos que se llaman entre sí directa o indirectamente.

Nota: Si recibe un error de compilación que indica que `CALC_RATING` no está definido, sólo debe considerarlo un problema si `CALC_RATING` es un procedimiento empaquetado privado. Si `CALC_RATING` está declarado en la especificación del paquete, el compilador resuelve la referencia del procedimiento público.

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;
/
```

Uso de Declaraciones Posteriores (continuación)

- La lista de parámetros formales debe aparecer en la declaración posterior y en el cuerpo del subprograma.
- El cuerpo del subprograma puede aparecer en cualquier lugar después de la declaración posterior, pero ambos deben aparecer en la misma unidad de programa.

Declaraciones Posteriores y Paquetes

Las declaraciones posteriores suelen permitir agrupar los subprogramas relacionados en un paquete. Las especificaciones de los subprogramas van en la especificación del paquete y los cuerpos de los subprogramas en el cuerpo del paquete, donde son invisibles para las aplicaciones. De esta manera, los paquetes permiten ocultar los detalles de implementación.

Creación de un Procedimiento de un Solo Uso

```
CREATE OR REPLACE PACKAGE taxes
IS
    tax   NUMBER;
  ...  -- declara todos los procedimientos/funciones
públicas
END  taxes;
/
```

```
CREATE OR REPLACE PACKAGE BODY taxes
IS
  ... -- declara todas las variables privadas
  ... -- define los procedimientos/funciones
públicas/privadas
BEGIN
   SELECT   rate_value
   INTO     tax
   FROM     tax_rates
   WHERE    rate_name = 'TAX';
END taxes;
/
```

ORACLE

Definición de un Procedimiento de un Solo Uso Automático

Los procedimientos de un solo uso sólo se ejecutan una vez, es decir, la primera vez que se llama al paquete en la sesión de usuario. En la transparencia anterior, el valor actual de TAX se define con el valor de la tabla TAX_RATES la primera vez que se hace referencia al paquete TAXES.

Nota: Inicialice variables públicas o privadas con un procedimiento de un solo uso automático cuando la derivación sea demasiado complicada para embeberla en la declaración de la variable. En este caso, no inicialice la variable en la declaración, porque el procedimiento de un solo uso restablece el valor.

Al final de los procedimientos de un solo uso no se utiliza la palabra clave END. Observe que en el ejemplo de la transparencia no hay ningún END al final del procedimiento de un solo uso.

Restricciones en las Funciones de Paquetes Utilizadas en SQL

Una función que se llama desde:

- **Una consulta o una sentencia DML, no puede terminar la transacción actual, ni ni crear o realizar un rollback en un punto de grabación, ni modificar (ALTER) el sistema o la sesión.**
- **Una sentencia de consulta o una sentencia DML paralelizada, no puede ejecutar una sentencia DML ni modificar la base de datos.**
- **Una sentencia DML, no puede leer ni modificar la tabla que concretamente esa sentencia DML está modificando.**

Nota: No está permitido realizar llamadas a subprogramas que infringen las restricciones anteriores.

ORACLE

Control de los Efectos Secundarios

Para que Oracle Server ejecute una sentencia SQL que llama a una función almacenada, debe conocer el nivel de pureza de las funciones almacenadas, es decir, si las funciones carecen de efectos secundarios. Los efectos secundarios son cambios en las tablas de base de datos o en las variables empaquetadas públicas (las que están declaradas en la especificación del paquete). Los efectos secundarios pueden retrasar la ejecución de una consulta, devolver resultados dependientes de un orden (y, por lo tanto, indeterminados), o requerir que las variables de estado del paquete se mantengan entre sesiones de usuario. Hay varios efectos secundarios que no están permitidos cuando se llama a una función desde una consulta SQL o una sentencia DML. Por lo tanto, estas son las restricciones que se aplican a las funciones almacenadas que se llaman desde expresiones SQL:

- Una función que se llama desde una consulta o una sentencia DML no puede terminar la transacción actual, ni crear o realizar un rollback en un punto de grabación, ni modificar el sistema o la sesión.
- Una función que se llama desde una sentencia de consulta o una sentencia DML paralelizada no puede ejecutar una sentencia DML ni modificar la base de datos.
- Una función que se llama desde una sentencia DML no puede leer ni modificar la tabla que concretamente esa sentencia DML está modificando.

Control de los Efectos Secundarios (continuación)

Note: En las versiones anteriores a Oracle8*i*, la comprobación de pureza se solía realizar durante la compilación, incluyendo la directiva de compilador `PRAGMA RESTRICT_REFERENCES` en la especificación del paquete. No obstante, a partir de Oracle8*i*, las funciones escritas por el usuario se pueden llamar desde las sentencias SQL sin que se compruebe su pureza en tiempo de compilación. Puede utilizar la directiva `PRAGMA RESTRICT_REFERENCES` para solicitar al compilador PL/SQL que le verifique que una función sólo tiene los efectos secundarios que esperaba. Las sentencias SQL, los accesos de variables de paquetes o las llamadas a funciones que violan las restricciones declaradas, continúan emitiendo errores de compilación de PL/SQL para ayudarle a aislar el código que tiene tales efectos.

Nota: Las restricciones en las funciones que se han explicado son las mismas que se explicaron en la lección *"Creación de Funciones"*.

Paquete Definido por el Usuario: taxes_pack

```
CREATE OR REPLACE PACKAGE taxes_pack
IS
   FUNCTION tax (p_value IN NUMBER) RETURN NUMBER;
END taxes_pack;
/
```

Package created.

```
CREATE OR REPLACE PACKAGE BODY taxes_pack
IS
   FUNCTION tax (p_value IN NUMBER) RETURN NUMBER
   IS
    v_rate NUMBER := 0.08;
   BEGIN
     RETURN (p_value * v_rate);
   END tax;
END taxes_pack;
/
```

Package body created.

ORACLE

Ejemplo

Se encapsula la función TAX en el paquete TAXES_PACK. La función se llama desde sentencias SQL en bases de datos remotas.

Llamada a una Función de Paquete Definida por el Usuario desde una Sentencia SQL

```
SELECT taxes_pack.tax(salary), salary, last_name
FROM   employees;
```

TAXES_PACK.TAX(SALARY)	SALARY	LAST_NAME
1920	24000	King
1360	17000	Kochhar
1360	17000	De Haan
720	9000	Hunold
480	6000	Ernst
422.4	5280	Austin
422.4	5280	Pataballa
369.6	4620	Lorentz
960	12000	Greenberg

...

109 rows selected.

ORACLE

Llamadas a Funciones de Paquetes

Para llamar a las funciones PL/SQL hay que utilizar el mismo método que para llamar a las funciones SQL incorporadas.

Ejemplo

Se llama a la función TAX (del paquete TAXES_PACK) desde una sentencia SELECT.

Nota: Si utiliza versiones de Oracle anteriores a 8*i*, tendrá que afirmar el nivel de pureza de la función en la especificación del paquete utilizando PRAGMA RESTRICT_REFERENCES. Si no se especifica, recibirá un mensaje de error en el que se le indica que la función TAX no garantiza que la base de datos no se vaya a actualizar mientras llama a la función de paquete en una consulta.

Estado Persistente de las Variables de Paquetes: Ejemplo

```
CREATE OR REPLACE PACKAGE comm_package IS
  g_comm  NUMBER := 10;          --inicializado en 10
  PROCEDURE reset_comm (p_comm   IN   NUMBER);
END comm_package;
/
```

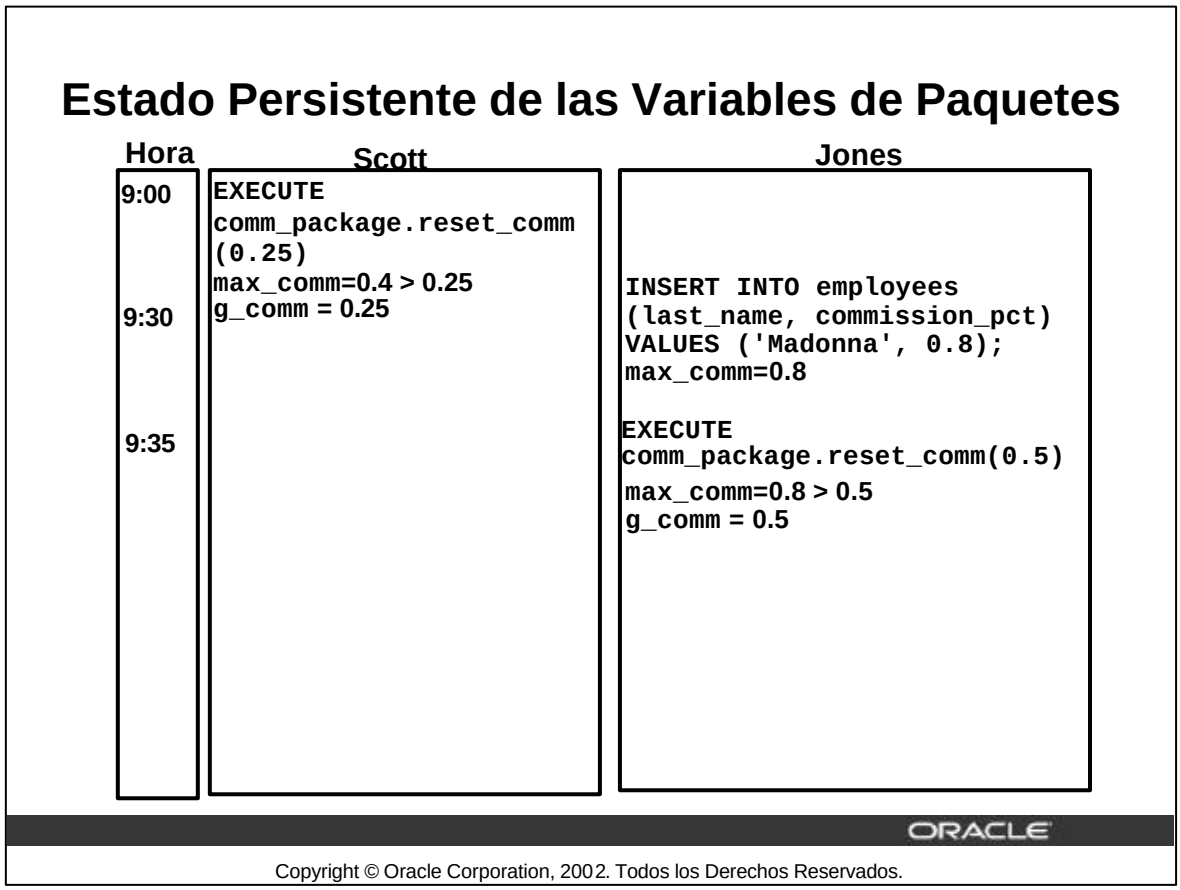
```
CREATE OR REPLACE PACKAGE BODY comm_package IS
   FUNCTION  validate_comm (p_comm  IN  NUMBER)
      RETURN BOOLEAN
   IS v_max_comm    NUMBER;
   BEGIN
    ...    -- valida que la comisión es menor que la comisión máxima
           -- comisión de la tabla
   END validate_comm;
   PROCEDURE  reset_comm (p_comm   IN  NUMBER)
   IS BEGIN
     ...   -- llama a validate_comm con el valor especificado
   END reset_comm;
END comm_package;
/
```

ORACLE

Estado Persistente de las Variables de Paquetes

Este paquete de ejemplo ilustra el estado persistente de las variables de paquetes. La función VALIDATE_COMM valida que la comisión no sea mayor que la comisión máxima que se gana actualmente. El procedimiento RESET_COMM llama a la función VALIDATE_COMM. Si intenta restablecer la comisión con el fin de que sea mayor que la comisión máxima, se provoca la excepción RAISE_APPLICATION_ERROR. En el ejemplo de la página siguiente se ha utilizado el procedimiento RESET_COMM.

Nota: Consulte en la página 13 de la lección 12 el código de la función VALIDATE_COMM y el procedimiento RESET_COMM. En la función VALIDATE_COMM, el sueldo máximo de la tabla EMPLOYEES se selecciona en la variable V_MAXSAL. Una vez que la variable tiene asignado un valor, este valor persiste en la sesión hasta que se vuelve a modificar. El ejemplo de la siguiente transparencia muestra cómo el valor de una variable de paquete global persiste durante una sesión.



Control del Estado Persistente de una Variable de Paquete

Puede realizar un seguimiento del estado de una variable o de un cursor de paquete, que persiste durante la sesión de usuario, desde el momento en que el usuario hace referencia por primera vez a la variable o al cursor hasta que dicho usuario se desconecta.

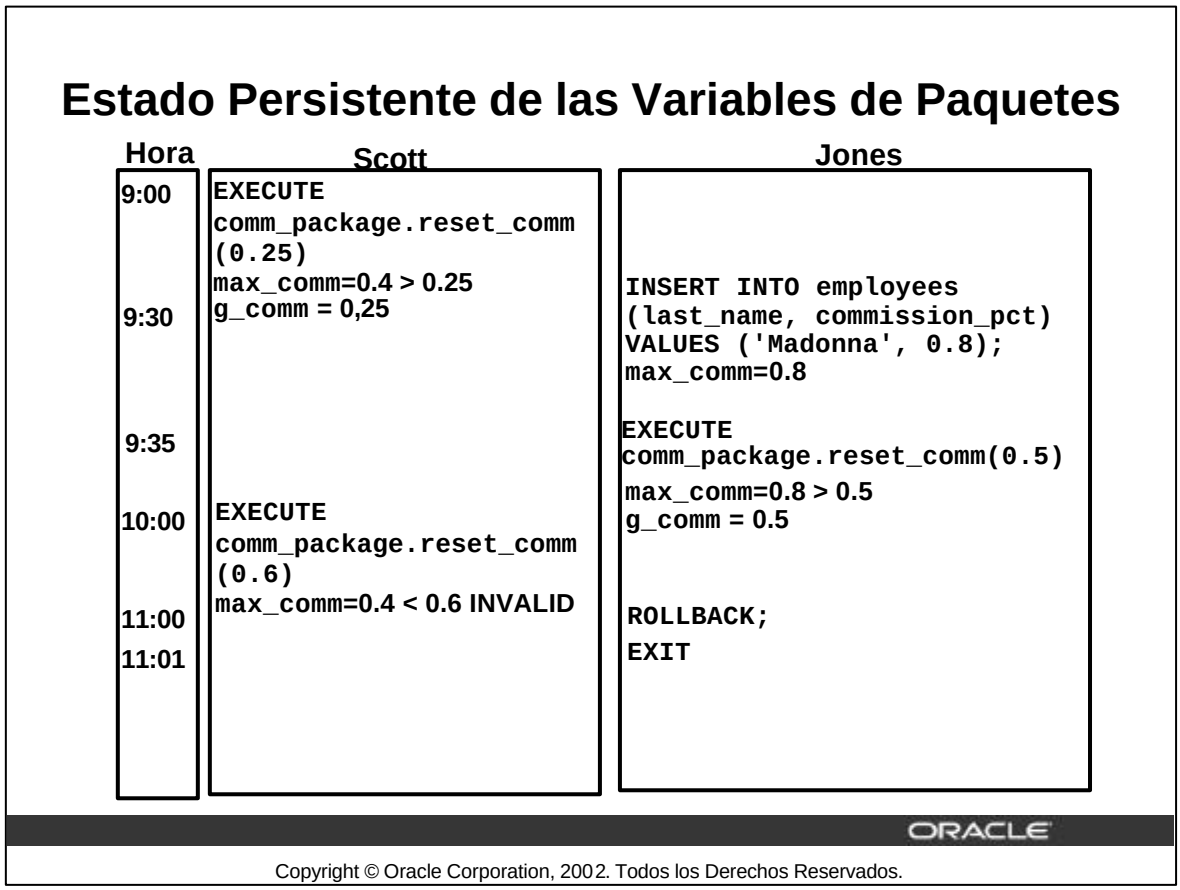
1. Inicialice la variable en su declaración o en un procedimiento de un solo uso automático.
2. Cambie el valor de la variable por medio de procedimientos de paquetes.
3. El valor de la variable se libera cuando el usuario se desconecta.

La secuencia de pasos de la transparencia muestra cómo persiste el estado de una variable de paquete.

9:00: Cuando Scott llama al procedimiento `RESET_COMM` con un porcentaje de comisión de 0.25, la variable global `G_COMM` se inicializa en 10 en su sesión. Se valida el valor 0.25 con el porcentaje de comisión máximo de 0.4 (obtenido de la tabla `EMPLOYEES`). Dado que 0.25 es menor que 0.4, la variable global se define en 0.25.

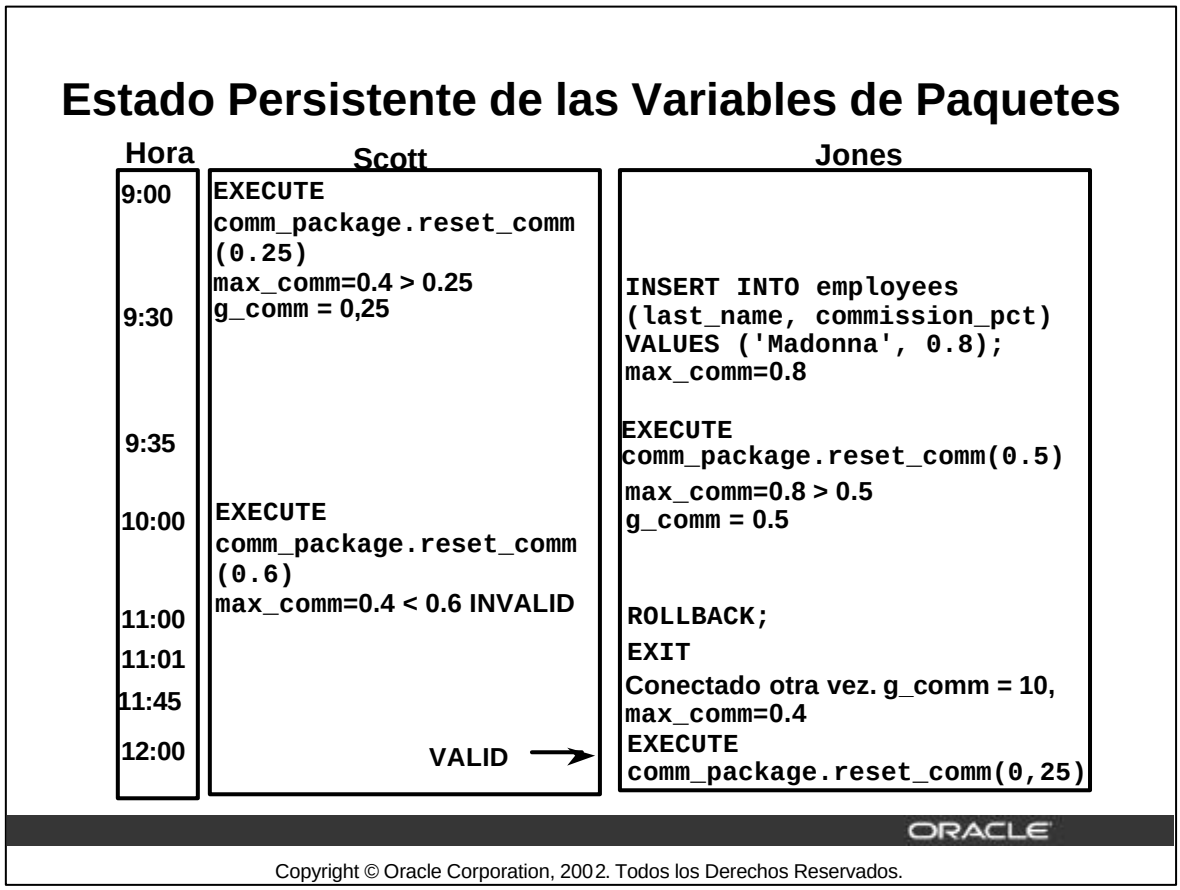
9:30: Jones inserta una nueva fila en la tabla `EMPLOYEES` con un porcentaje de comisión de 0.8.

9:35: Jones llama al procedimiento `RESET_COMM` con un porcentaje de comisión de 0.5. La variable `G_COMM` se inicializa en 10 en esta sesión. Se valida el valor 0.5 con el porcentaje de comisión máximo de 0.8 (porque la nueva fila tenía el valor 0.8). Dado que 0.5 es menor que 0.8, la variable global se define en 0.5.



Control del Estado Persistente de una Variable de Paquete (continuación)

10:00: Scott llama al procedimiento con un porcentaje de comisión de 0.6. Este valor es superior al porcentaje de comisión máximo de 0.4 (Scott no podía ver el valor porque Jones no había completado la transacción). Por lo tanto, no es válido.



Control del Estado Persistente de una Variable de Paquete (continuación)

11:00 a 12:00: Jones realiza un rollback en la transacción y sale de la sesión. El valor global se inicializa en 10 cuando se conecta a las 11:45. El procedimiento se realiza con éxito porque el nuevo valor, 0.25, es menor que el valor máximo de 0.4.

Control del Estado Persistente de un Cursor de Paquete

Ejemplo:

```
CREATE OR REPLACE PACKAGE pack_cur
IS
 CURSOR c1 IS  SELECT employee_id
               FROM  employees
               ORDER BY employee_id DESC;
 PROCEDURE proc1_3rows;
PROCEDURE proc4_6rows;
END pack_cur;
/
```

```
Package created.
```

ORACLE

Control del Estado Persistente de un Cursor de Paquete

Ejemplo

Siga estos pasos para controlar un cursor público:

1. Declare el cursor público (global) en la especificación del paquete.
2. Abra el cursor y recupere filas sucesivas desde el cursor utilizando un procedimiento empaquetado (público), `PROC1_3ROWS`.
3. Continúe recuperando filas sucesivas desde el cursor y luego ciérrelo utilizando otro procedimiento empaquetado (público), `PROC4_6ROWS`.

La transparencia muestra la especificación del paquete `PACK_CUR`.

Control del Estado Persistente de un Cursor de Paquete

```
CREATE OR REPLACE PACKAGE BODY pack_cur  IS
  v_empno NUMBER;
  PROCEDURE proc1_3rows IS
  BEGIN
    OPEN c1;
    LOOP
     FETCH c1 INTO v_empno;
     DBMS_OUTPUT.PUT_LINE('Id :' ||(v_empno));
     EXIT WHEN c1%ROWCOUNT >= 3;
    END LOOP;
  END proc1_3rows;
  PROCEDURE proc4_6rows IS
  BEGIN
    LOOP
     FETCH c1 INTO v_empno;
     DBMS_OUTPUT.PUT_LINE('Id :' ||(v_empno));
     EXIT WHEN c1%ROWCOUNT >= 6;
    END LOOP;
    CLOSE c1;
  END proc4_6rows;
END pack_cur;
/
```

ORACLE

Control del Estado Persistente de un Cursor de Paquete (continuación)

Ejemplo

La transparencia de esta página muestra el cuerpo del paquete PACK_CUR que sirve de apoyo a la especificación del paquete. En el cuerpo del paquete:

1. Abra el cursor y recupere filas sucesivas desde el cursor utilizando un procedimiento empaquetado, PROC1_3ROWS.
2. Continúe recuperando filas sucesivas desde el cursor y luego ciérrelo utilizando otro procedimiento empaquetado, PROC4_6ROWS.

Ejecución de PACK_CUR

```
SET SERVEROUTPUT ON
EXECUTE pack_cur.proc1_3rows
EXECUTE pack_cur.proc4_6rows
```

```
Id :208
Id :207
Id :206
PL/SQL procedure successfully completed.
Id :205
Id :204
Id :203
PL/SQL procedure successfully completed.
```

ORACLE

Resultado de la Ejecución de PACK_CUR

El estado de la variable o el cursor de paquete persiste en las transacciones de una sesión. El estado no persiste de una sesión a otra de un mismo usuario, ni tampoco de usuarios distintos.

Tablas y Registros PL/SQL en Paquetes

```
CREATE OR REPLACE PACKAGE emp_package IS
  TYPE emp_table_type IS TABLE OF employees%ROWTYPE
     INDEX BY BINARY_INTEGER;
  PROCEDURE read_emp_table
              (p_emp_table OUT emp_table_type);
END emp_package;
/
```

```
CREATE OR REPLACE PACKAGE BODY emp_package IS
  PROCEDURE read_emp_table
              (p_emp_table OUT emp_table_type) IS
  i BINARY_INTEGER := 0;
  BEGIN
    FOR emp_record IN (SELECT * FROM employees)
    LOOP
      p_emp_table(i) := emp_record;
      i:= i+1;
    END LOOP;
  END read_emp_table;
END emp_package;
/
```

ORACLE

Transferencia de Tablas de Registros a Procedimientos o Funciones de un Paquete

Se llama al procedimiento READ_EMP_TABLE desde un bloque PL/SQL anónimo, utilizando *i*SQL*Plus.

```
DECLARE
  v_emp_table emp_package.emp_table_type;
  BEGIN
    emp_package.read_emp_table(v_emp_table);
    DBMS_OUTPUT.PUT_LINE('An example: '||v_emp_table(4).last_name);
END;
/
```

```
An example: Ernst
PL/SQL procedure successfully completed.
```

Para llamar al procedimiento READ_EMP_TABLE desde otro procedimiento o desde cualquier bloque PL/SQL en el exterior del paquete, el parámetro real que hace referencia al parámetro OUT P_EMP_TABLE debe estar prefijado con el nombre de su paquete. En el ejemplo anterior, se declara la variable V_EMP_TABLE del tipo EMP_TABLE_TYPE con el nombre del paquete agregado como prefijo.

Resumen

En esta lección, ha aprendido a:

- **Sobrecargar subprogramas**
- **Utilizar referencias posteriores**
- **Utilizar procedimientos de un solo uso**
- **Describir el nivel de pureza de las funciones de paquetes**
- **Identificar el estado persistente de los objetos empaquetados**

ORACLE

Resumen

La sobrecarga es una función que permite definir diferentes subprogramas con el mismo nombre. Es lógico que dos subprogramas tengan el mismo nombre si el procesamiento de ambos es igual y sólo varían los parámetros que se les transfieren.

PL/SQL permite utilizar una declaración de subprograma especial que se denomina declaración posterior. La declaración posterior permite definir subprogramas en orden lógico o alfabético, definir subprogramas recursivos y agrupar subprogramas en un paquete.

Los procedimientos de un solo uso sólo se ejecutan la primera vez que se llama al paquete en la sesión del otro usuario. Puede utilizar esta función para inicializar variables sólo una vez por sesión.

Puede realizar un seguimiento del estado de una variable o de un cursor de paquete, que persiste durante la sesión de usuario, desde el momento en que el usuario hace referencia por primera vez a la variable o el cursor hasta que éste se desconecta.

Visión General de la Práctica 13

Esta práctica cubre los siguientes temas:

- **Uso de subprogramas sobrecargados**
- **Creación de un procedimiento de un solo uso**

ORACLE

Visión General de la Práctica 13

En esta práctica deberá crear un paquete que contiene una función sobrecargada. También tendrá que crear un procedimiento de un solo uso en el interior de un paquete para rellenar una tabla PL/SQL.

Práctica 13

1. Cree un paquete llamado OVER_LOAD. Cree dos funciones en este paquete y llame a cada una de ellas PRINT_IT. La función acepta una fecha o una cadena de caracteres e imprime una fecha o un número, dependiendo de cómo se llame a la función.

   Nota:

   - Para imprimir el valor de la fecha, utilice el formato de entrada DD-MON-YY y el formato de salida FmMonth,dd yyyy. Asegúrese de manejar entradas no válidas.
   - Para imprimir el número, utilice el formato de entrada 999,999.00.

   a. Pruebe la primera versión de PRINT_IT con el siguiente conjunto de comandos:

```
VARIABLE display_date VARCHAR2(20)
EXECUTE :display_date := over_load.print_it (TO_DATE('08-MAR-01'))
PRINT display_date
```

PL/SQL procedure successfully completed.

DISPLAY_DATE
March,8 2001

   b. Pruebe la segunda versión de PRINT_IT con el siguiente conjunto de comandos:

```
VARIABLE g_emp_sal NUMBER
EXECUTE :g_emp_sal := over_load.print_it('33,600')
PRINT g_emp_sal
```

PL/SQL procedure successfully completed.

G_EMP_SAL
33600

2. Cree un nuevo paquete, llamado CHECK_PACK, para implementar una nueva regla de negocio.

   a. Cree un procedimiento denominado CHK_DEPT_JOB para verificar si una combinación determinada de identificador de departamento y puesto de trabajo es válida. En este caso, por *válida* se entiende una combinación que exista actualmente en la tabla EMPLOYEES.

      Nota:

      - Utilice una tabla PL/SQL para almacenar la combinación válida de departamento y puesto de trabajo.
      - Sólo hay que rellenar una vez la tabla PL/SQL.
      - Provoque un error de aplicación con el mensaje adecuado si la combinación no es válida.

   b. Pruebe el procedimiento de paquete CHK_DEPT_JOB ejecutando el siguiente comando:

```
EXECUTE check_pack.chk_dept_job(50,'ST_CLERK')
```

      ¿Qué ocurre?

   c. Pruebe el procedimiento de paquete CHK_DEPT_JOB ejecutando el siguiente comando:

```
EXECUTE check_pack.chk_dept_job(20,'ST_CLERK')
```

      ¿Qué ocurre y por qué?