



# Paquetes (Packages)

# Definición

Los paquetes (“Packages”) son agrupaciones de procedimientos, funciones, variables y constantes que pueden ser accedidas por las diferentes aplicaciones

# Declaración de un Paquete

- La declaración de un paquete tiene 2 partes que se almacenan separadamente en la BD:
  - Una especificación de paquete (**specification**), que es obligatoria
  - Un cuerpo (**body**), el cual es opcional

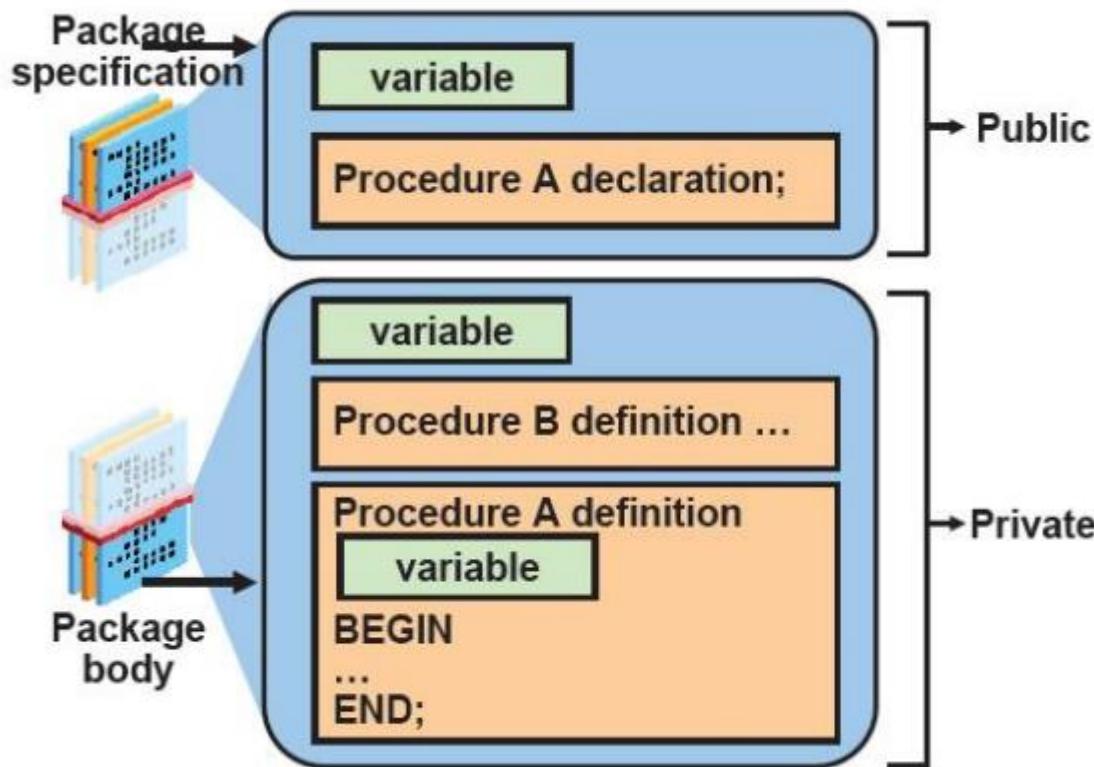
# Especificación del Paquete (Package Specification)

- Es la “interfaz” para las aplicaciones. Contiene las declaraciones de procedimientos, funciones, variables que podrán ser accedidos desde afuera del paquete
- La especificación puede también incluir PRAGMAS, que son directivas para el compilador
- Los componentes definidos en esta sección son de acceso público

# Cuerpo del Paquete (Package Body)

- Es el desarrollo de los procedimientos y funciones que fueron previamente declarados en la especificación del paquete.
- En esta parte, a su vez se pueden declarar otras variables, procedimientos y funciones que ya no serán accesibles desde fuera del paquete (**componentes privados**)

# Paquetes – Representación Gráfica



# Creación de la Especificación del Paquete

```
CREATE [ OR REPLACE ] PACKAGE
<nombre> [declaración de
derechos del invocador]
IS|AS
    definición de tipo
    especificación de procedimientos o
    funciones
    declaración de variables
    declaración de cursores
END [<nombre>];
```

# Creación del Cuerpo del Paquete

```
CREATE [ OR REPLACE ] PACKAGE BODY
<nombre>
IS | AS
declaración de variables, cursores y
tipos locales
desarrollo de los procedimientos
especificados
END [<nombre>] ;
```

# Ejemplo

## Creación de la Especificación

```
CREATE OR REPLACE PACKAGE ventas_pkg AS  
    PROCEDURE act_stock (p_idart number, pcantidad  
number);  
    FUNCTION f_stock_act (p_idart number) RETURN  
number;  
END;  
/
```

# Creación del Cuerpo

```
CREATE OR REPLACE PACKAGE BODY ventas_pkg IS
    PROCEDURE act_stock (p_idart number, pcantidad
    number) IS
        BEGIN
            UPDATE b_articulos
                SET stock_actual= stock_actual - pcantidad
            WHERE id = p_idart;
        END;
    FUNCTION f_stock_act (p_idart number) RETURN
    number IS
        v_stock b_articulos.stock_actual%TYPE;
        BEGIN
            SELECT a.stock_actual INTO v_stock
            FROM b_articulos a
            WHERE a.id=p_idart;
            RETURN v_stock;
        EXCEPTION
            WHEN no_data_found THEN RETURN null;
        END;
    END;
/
```

# Para eliminar un paquete

- Para eliminar la especificación del paquete:
  - `DROP PACKAGE <nombre_paquete>;`
- Para eliminar el cuerpo del paquete
  - `DROP PACKAGE BODY <nombre_paquete>;`

# Para ver los paquetes en el diccionario

- Para ver si están los paquetes y si son válidos o no:

```
SELECT OBJECT_NAME, STATUS, CREATED  
FROM USER_OBJECTS  
WHERE OBJECT_TYPE = 'PACKAGE';
```

```
SELECT OBJECT_NAME, STATUS, CREATED  
FROM USER_OBJECTS  
WHERE OBJECT_TYPE = 'PACKAGE BODY';
```

# Para ver los paquetes en el diccionario:

- Para ver el código de los paquetes:

```
SELECT text  
FROM user_source  
WHERE name = 'VENTAS_PKG' AND type = 'PACKAGE';
```

```
SELECT text  
FROM user_source  
WHERE name = 'VENTAS_PKG' AND type = 'PACKAGE  
BODY';
```

# Consideraciones para escribir paquetes

- Construya paquetes para uso general de una aplicación
- Siempre debe crear la especificación del paquete antes que el cuerpo
- La especificación sólo debe contener los objetos que se desea que sean públicos
- Debe tener en cuenta que si cambia el paquete (agrega o quita componentes), se requiere recompilación de los programas que lo referencian (éstos quedan inválidos)



# • PACKAGES

## Consideraciones Adicionales

# Sobrecarga de funciones

- Los procedimientos y funciones pueden sobrecargarse en un paquete
- Es decir, puede haber más de un procedimiento o función con el mismo nombre y diferentes parámetros
- No se puede sobrecargar subprogramas si los parámetros solo varían en nombre o modo
- No se puede sobrecargar subprogramas si solo varían en el tipo de datos que devuelven

# Ejemplo de Sobrecarga:

```
CREATE OR REPLACE PACKAGE emp_pkg AS
  FUNCTION f_obtener_salario(p_cedula number)
return number;
  FUNCTION f_obtener_salario(p_apellido
varchar2, p_nombre varchar2) RETURN number;
END;
/
```

## Declaración de derechos del invocador

```
CREATE [ OR REPLACE ] PACKAGE
<nombre> [<AUTHID { CURRENT_USER
| DEFINER } >] IS | AS
<declaraciones;>
END [<nombre>];
```

La cláusula AUTHID afecta la manera como Oracle resuelve la verificación de los objetos y privilegios en tiempo de ejecución.

Puede ser: DEFINER o CURRENT\_USER

## Declaración de derechos del invocador

**AUTHID CURRENT\_USER:** indica que el paquete se ejecutará con los privilegios del CURRENT USER (usuario actual).

**AUTHID DEFINER:** indica que el paquete se ejecutará con los privilegios del propietario (owner) del esquema en el cual reside el paquete, por lo que los nombres externos se resolverán en el esquema que el paquete reside.

# Dependencia de los subprogramas

- Cuando se compila una función o un procedimiento almacenado, todos los objetos de Oracle a los que hace referencia se registran en el diccionario de datos
- Si se alteran algunos de estos objetos (ej. Una tabla) a los que hace referencia un subprograma, éste puede quedar inválido

# En los paquetes

- El cuerpo del paquete depende de su especificación (cabecera) y de los objetos a los que hace referencia
- Si se cambia la cabecera se invalida el cuerpo
- Si se cambia un objeto al que hace referencia algún procedimiento del body, solo éste queda invalidado, no así la cabecera

# Para poder ver las dependencias

- **USER\_DEPENDENCIES**
- **ALL\_DEPENDENCIES**
- **DBA\_DEPENDENCIES**

## Ver las dependencias

```
SELECT NAME, TYPE, REFERENCED_NAME, REFERENCED_TYPE  
FROM USER_DEPENDENCIES;
```

### Ejemplo:

#### Ver todos los objetos que dependen de la tabla 'b\_personas'

```
SELECT name, type  
FROM user_dependencies  
WHERE referenced_name = 'B_PERSONAS';
```



- **CONSIDERACIONES  
SOBRE EL USO DE  
MEMORIA**

# El área global (SGA)

- El área denominada SHARED POOL en el SGA, se usa para mantener el formato compilado de las unidades de programa almacenadas. Cuanto más grande, mayor el área de memoria utilizada
- Cuando se invoca un subprograma de un paquete (incluso si se invoca una variable de la cabecera), la versión compilada del paquete completo se carga en el SHARED POOL

# El área global (SGA)

- Si otro usuario invoca la misma unidad de programa, ya estará residente en memoria, hasta que el SGA se llene. Si esto pasa, Oracle libera espacio usando el algoritmo LRU (least recently used)
- Si se usa frecuentemente esto es menos probable que suceda.

# Para optimizar el uso de memoria

- Crear paquetes que contienen subprogramas relacionados lógicamente para un área de negocio en particular
- Reservar buen espacio para un almacenamiento en el área de memoria compartida

(SHARED\_POOL\_RESERVED\_SIZE al menos 10% del SHARED\_POOL\_SIZE)

# Uso del Pragma serially\_reusable

- Las variables y otros objetos de los paquetes consumen memoria que corresponde a la sesión del usuario y permanecen durante toda la sesión aunque ya no se utilicen
- Usando el PRAGMA SERIALLY\_REUSABLE, el paquete no se guarda en cada área de memoria del usuario, sino en un solo lugar de la memoria global, permitiendo el uso por diferentes usuarios
- Por lo tanto, al terminar el uso, el área es liberada



# **OCULTANDO EL CODIGO INTERNO DEL PAQUETE: PL/SQL WRAPPER**

# Wrapper

- Es un utilitario del ORACLE que permite ocultar el texto de un paquete convirtiendo el código fuente en un código objeto portable e independiente de la plataforma
- Sólo el cuerpo del paquete debe ser empaquetado
- SINTAXIS:

```
WRAP INAME=archivo_entrada  
ONAME=archivo_salida
```

# Wrapper - Ejemplo

## --Archivo pack\_s.sql

```
CREATE OR REPLACE PACKAGE P_PACK
AS
    FUNCTION F_REMP (N NUMBER) RETURN BOOLEAN;
END;
```

## --Archivo pack\_b.sql

```
CREATE OR REPLACE PACKAGE BODY P_PACK
AS
    FUNCTION F_REMP (N NUMBER) RETURN BOOLEAN
    IS
        BEGIN
            IF N = 1 THEN
                RETURN TRUE;
            ELSE
                RETURN FALSE;
            END IF;
        END;
    END;
```

# Wrapper - Ejemplo

```
SQL> HOST WRAP  
INAME=c:\cnmw\sql\pack_b.sql  
ONAME=c:\cnmw\sql\pack_b.plb
```

```
SQL> START c:\cnmw\sql\pack_b.plb
```