**Índice**

* [Permite modularizar el diseño de nuestra aplicación](https://elbauldelprogramador.com/plsql-paquetes-packages/#permite-modularizar-el-dise%C3%B1o-de-nuestra-aplicaci%C3%B3n)
* [Otorga flexibilidad al momento de diseñar la aplicación](https://elbauldelprogramador.com/plsql-paquetes-packages/#otorga-flexibilidad-al-momento-de-dise%C3%B1ar-la-aplicaci%C3%B3n)
* [Permite ocultar los detalles de implementación](https://elbauldelprogramador.com/plsql-paquetes-packages/#permite-ocultar-los-detalles-de-implementaci%C3%B3n)
* [Agrega mayor funcionalidad a nuestro desarrollo](https://elbauldelprogramador.com/plsql-paquetes-packages/#agrega-mayor-funcionalidad-a-nuestro-desarrollo)
* [Introduce mejoras al rendimiento](https://elbauldelprogramador.com/plsql-paquetes-packages/#introduce-mejoras-al-rendimiento)
* [Permite la “Sobrecarga de funciones” (Overloading).](https://elbauldelprogramador.com/plsql-paquetes-packages/#permite-la-sobrecarga-de-funciones-overloading)

Además de brindarnos múltiples elementos que nos permiten desarrollar una aplicación robusta, Oracle nos ofrece la posibilidad de programar en forma modular, clara y eficiente. En este apartado veremos cómo embeber [procedimientos, funciones](https://elbauldelprogramador.com/plsql-procedimientos-y-funciones/), definiciones de tipos de datos y [declaraciones de variables](https://elbauldelprogramador.com/plsql-declaracion-de-variables/) en una misma estructura que los agrupe y relacione lógicamente. Esta estructura se denomina Package (Paquete) y su uso nos permite no sólo mejorar la calidad de diseño de nuestras aplicaciones sino también optimizar el desempeño de las mismas.

**Definición**

Un Paquete es un objeto PL/Sql que agrupa lógicamente otros objetos PL/Sql relacionados entre sí, encapsulándolos y convirtiéndolos en una unidad dentro de la base de datos.

Los Paquetes están divididos en 2 partes: especificación (obligatoria) y cuerpo (no obligatoria). La especificación o encabezado es la interfaz entre el Paquete y las aplicaciones que lo utilizan y es allí donde se declaran los tipos, variables, constantes, [excepciones](https://elbauldelprogramador.com/plsql-excepciones/), [cursores](https://elbauldelprogramador.com/plsql-cursores/), procedimientos y funciones que podrán ser invocados desde fuera del paquete.

En el cuerpo del paquete se implementa la especificación del mismo. El cuerpo contiene los detalles de implementación y declaraciones privadas, manteniéndose todo esto oculto a las aplicaciones externas, siguiendo el conocido concepto de “caja negra”. Sólo las declaraciones hechas en la especificación del paquete son visibles y accesibles desde fuera del paquete (por otras aplicaciones o procedimientos almacenados) quedando los detalles de implementación del cuerpo del paquete totalmente ocultos e inaccesibles para el exterior.

Para acceder a los elementos declarados en un paquete basta con anteceder el nombre del objeto referenciado con el nombre del paquete donde está declarado y un punto, de esta manera: **Paquete.Objeto** donde Objeto puede ser un tipo, una variable, un cursor, un procedimiento o una función declarados dentro del paquete.

Para referenciar objetos desde adentro del mismo paquete donde han sido declarados no es necesario especificar el nombre del paquete y pueden (deberían) ser referenciados directamente por su nombre. Finalmente y siguiendo a la parte declarativa del cuerpo de un paquete puede, opcionalmente, incluirse la sección de inicialización del paquete. En esta sección pueden, por ejemplo, inicializarse variables que utiliza el paquete. La sección de inicialización se ejecuta sólo la primera vez que una aplicación referencia a un paquete, esto es, se ejecuta sólo una vez por sesión.

Como hemos dicho anteriormente, la creación de un paquete pasa por dos fases:

* Crear la cabecera del paquete donde se definen que procedimientos, funciones, variables, cursores, etc. Disponibles para su uso posterior fuera del paquete. En esta parte solo se declaran los objetos, no se implementa el código.
* Crear el cuerpo del paquete, donde se definen los bloques de código de las funciones y procedimientos definidos en la cabecera del paquete.

Para crear la cabecera del paquete utilizaremos la siguiente instrucción:

**CREATE** {**OR** **REPLACE**} PACKAGE nombre\_de\_paquete **IS**

*-- Declaraciones*

**END**;

Para crear el cuerpo del paquete utilizaremos la siguiente instrucción:

**CREATE** {**OR** **REPLACE**} PACKAGE BODY nombre\_paquete **IS**

*--Bloques de código*

**END**;

Hay que tener en cuenta que toda declaración de función o procedimiento debe estar dentro del cuerpo del paquete, y que todo bloque de código contenido dentro del cuerpo debe estar declarado dentro de la cabecera de paquete.

Cuando se quiera acceder a las funciones, procedimientos y variables de un paquete se debe anteponer el nombre de este:

Nombre\_paquete.función(x);

Nombre\_paquete.procedimiento(x);

Nombre\_paquete.**variable**;

**CREATE** **OR** **REPLACE** PACKAGE PK1 **IS**

**PROCEDURE** xLis (xfamilia **IN** Articulos.cArtFml**%TYPE**);

**END**;

**CREATE** **OR** **REPLACE** PACKAGE BODY PK1 **IS**

**procedure** xLis (xfamilia Articulos.cArtCdg**%TYPE**)

**IS**

xfam Articulos.cArtFml**%type**;

xCod Articulos.cArtCdg**%TYPE**;

xDes Articulos.cArtDsc**%TYPE**;

**CURSOR** cArticulos **IS** **SELECT** cArtCdg,cArtDsc **FROM** Articulos

**WHERE** cArtFml **=** xfam;

**BEGIN**

xfam :**=** xfamilia;

**OPEN** cArticulos;

LOOP

**FETCH** cArticulos **INTO** xCod,xDes;

EXIT **WHEN** cArticulos**%**NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE (xDes);

**END** LOOP;

**CLOSE** cArticulos;

**END**;

**END**;

**Ventajas del uso de Paquetes**

Dentro de las ventajas que ofrece el uso de paquetes podemos citar que:

**Permite modularizar el diseño de nuestra aplicación**

El uso de Paquetes permite encapsular elementos relacionados entre sí (tipos, variables, procedimientos, funciones) en un único módulo PL/Sql que llevará un nombre que identifique la funcionalidad del conjunto.

**Otorga flexibilidad al momento de diseñar la aplicación**

En el momento de diseñar una aplicación todo lo que necesitaremos inicialmente es la información de interfaz en la especificación del paquete. Puede codificarse y compilarse la especificación sin su cuerpo para posibilitar que otros sub-programas que referencian a estos elementos declarados puedan compilarse sin errores. De esta manera podremos armar un “prototipo” de nuestro sistema antes de codificar el detalle del mismo.

**Permite ocultar los detalles de implementación**

Pueden especificarse cuáles tipos, variables y sub-programas dentro del paquete son públicos (visibles y accesibles por otras aplicaciones y sub-programas fuera del paquete) o privados (ocultos e inaccesibles fuera del paquete). Por ejemplo, dentro del paquete pueden existir procedimientos y funciones que serán invocados por otros programas, así como también otras rutinas de uso interno del paquete que no tendrán posibilidad de ser accedidas fuera del mismo. Esto asegura que cualquier cambio en la definición de estas rutinas internas afectará sólo al paquete donde se encuentran, simplificando el mantenimiento y protegiendo la integridad del conjunto.

**Agrega mayor funcionalidad a nuestro desarrollo**

Las definiciones públicas de tipos, variables y cursores hechas en la especificación de un paquete persisten a lo largo de una sesión. Por lo tanto pueden ser compartidas por todos los sub-programas y/o paquetes que se ejecutan en ese entorno durante esa sesión. Por ejemplo, puede utilizarse esta técnica (en dónde sólo se define una especificación de paquete y no un cuerpo) para mantener tipos y variables globales a todo el sistema.

**Introduce mejoras al rendimiento**

En relación a su ejecución, cuando un procedimiento o función que está definido dentro de un paquete es llamado por primera vez, todo el paquete es ingresado a memoria. Por lo tanto, posteriores llamadas al mismo u otros sub-programas dentro de ese paquete realizarán un acceso a memoria en lugar de a disco. Esto no sucede con procedimientos y funciones estándares.

**Permite la “Sobrecarga de funciones” (Overloading).**

PL/SQL nos permite que varios procedimientos o funciones almacenadas, declaradas dentro de un mismo paquete, tengan el mismo nombre. Esto nos es muy útil cuando necesitamos que los sub-programas puedan aceptar parámetros que contengan diferentes tipos de datos en diferentes instancias. En este caso Oracle ejecutará la “versión” de la función o procedimiento cuyo encabezado se corresponda con la lista de parámetros recibidos.

**Declaración de cursores en paquetes**

En los paquetes también podemos [introducir cursores](https://desarrolloweb.com/articulos/cursores-pl-sql-I.html), para ello debemos declararlo en la cabecera del paquete indicando su nombre, los parámetros y tipo devuelto. Para que lo veáis más claro os dejo un ejemplo a continuación:

CREATE or REPLACE PACKAGE empleados AS  
   .....  
   CURSOR a RETURN empleado%ROWTYPE;  
   ...  
   END empleados;  
  
CREATE or REPLACE PACKAGE BODY empleados AS  
   ....  
   CURSOR a RETURN empleado%ROWTYPE  
      SELECT \* FROM empleado WHERE salario < 10000;  
   ....  
END empleados;

Los paquetes suministrados por Oracle son:

Standard : tiene la función to\_char y abs  
dbms\_standard: tiene la función raise\_aplication\_error  
dbms\_output: con la función put\_line  
dbms\_sql: que utiliza sql de forma dinámica.

**Utilización de los objetos definidos en los paquetes**

Podemos utilizar los objetos definidos en los paquetes básicamente de dos maneras distintas:

Desde el mismo paquete: esto quiere decir que cualquier objeto puede ser utilizado dentro del paquete por otro objeto declarado en el mismo.  
Para utilizar un objeto dentro del paquete tan sólo tendríamos que llamarlo. La llamada sería algo así: v\_emple :=buscar\_emple.datos(v\_n\_ape); (como veis no utilizamos el execute ya que nos encontramos dentro del paquete).

Desde fuera del paquete: Podemos utilizar los objetos de un paquete siempre y cuando haya sido declarado en la especificación del mismo. Para llamar a un objeto o procedimiento desde fuera del paquete utilizaríamos la siguiente notación: execute nombre\_paquete.nombre\_procedimiento(lista de

   ......