**PAQUETES**

Un paquete es una utilidad de Oracle. Por sí mismo no sirve de nada.

Un paquete es una estructura que contiene objetos lógicos en su interior.

Sirve para agrupar objetos que tengan que ver entre sí para organizar la bbdd.

Como objetos lógicos, hablamos de procedimientos y funciones.

Imaginemos que tenemos varios procedimientos que realizan acciones sobre una misma tabla (EMP). Insertamos, eliminamos, modificamos y mostramos registros.

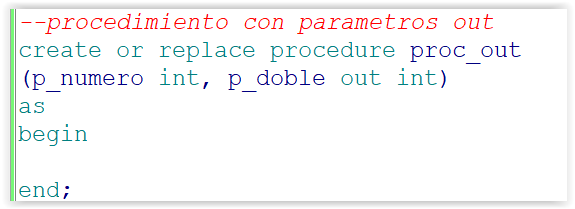
En lugar de tener cada procedimiento suelto, lo que se hace es agruparlo dentro de un objeto Paquete, como una carpeta.

Dentro de dicho paquete estarán todos los procedimientos.

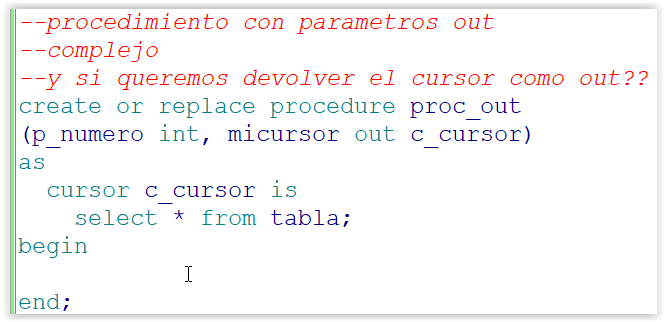
El paquete, aunque no vamos a verlo tan a fondo hasta Java, también permite la posibilidad de poder declarar variables para poder recuperarlas en la salida.

Un paquete también puede estar formado solamente por declaraciones, lo que es una ventaja para poder utilizar parámetros OUT compuestos.

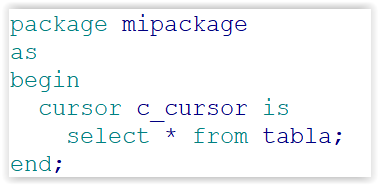
Este procedimiento tiene un tipo simple (INT)



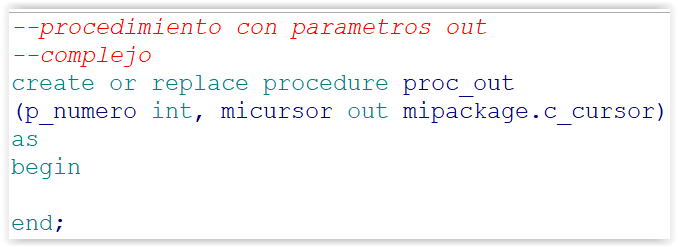
Procedimiento con un tipo complejo (Un out de un cursor) no se puede hacer sin paquetes.



Lo que se hace es crear un paquete con una declaración de tipos complejos, como cursores, por ejemplo.



Posteriormente se llama desde el procedimiento a la declaración de los cursores del paquete.



Un paquete está dividido en dos bloques PL-SQL.

**El paquete debe tener el mismo nombre en los dos bloques.**

Se ejecuta cada bloque de forma individual.

1. **Cabecera**: Es la declaración de elementos que van a formar parte del paquete.

Si un paquete es de solo declaraciones (solo declaración de cursores) solamente será necesaria la cabecera.

Sintaxis:

create or replace package nombrepaquete

as

declaración de contenido del paquete

end [nombrepaquete];

1. **Cuerpo**: El cuerpo del paquete es la implementación de los elementos lógicos declarados. Implementación de procedimientos o funciones.

Si el paquete tiene procedimientos o funciones, es obligatorio crear el cuerpo.

Sintaxis:

create or replace package **body** nombrepaquete

as

Implementación de procedures o functions del paquete

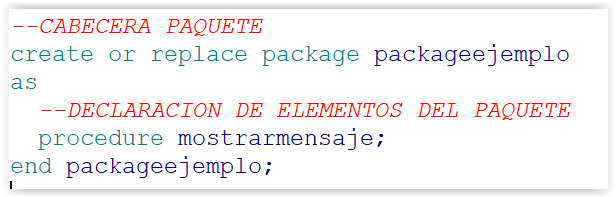
end [nombrepaquete];

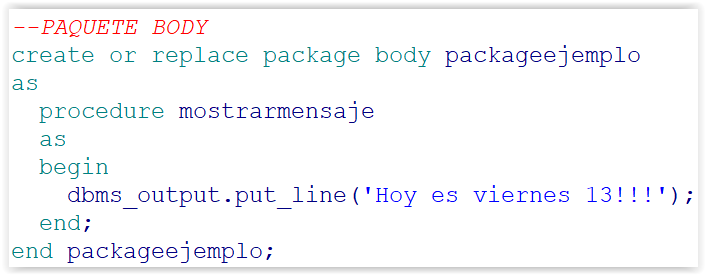
Para los dos componentes del paquete:

**Cuando declaramos funciones o procedimientos, ya no llevan create or replace.**

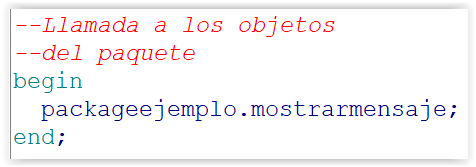
Si el paquete tiene una función, debemos indicar el tipo de dato que devolverá la función.

Ejemplo de un paquete con un procedimiento.





La llamada a los elementos de un paquete, solamente se diferencia en que tenemos que **incluir el nombre del paquete antes de los objetos.**



LOS PAQUETES SON ESTRUCTURAS PARA ORGANIZAR.

TODOS LOS PAQUETES TIENEN OBJETOS EN COMUN SOBRE UN TEMA

LA TEMATICA PUEDE SER UN CONJUNTO DE ACCIONES O TABLA.

PAQUETE QUE CONTENGA PROCEDIMIENTOS PARA **ELIMINAR**

SOBRE ALGUNAS TABLAS.

**create or replace package pk\_delete**

**AS**

**procedure eliminarempleado(p\_empno EMP.EMP\_NO%TYPE);**

**procedure eliminardepartamento(p\_deptno DEPT.DEPT\_NO%TYPE);**

**procedure eliminardoctor(p\_doctorno DOCTOR.DOCTOR\_NO%TYPE);**

**procedure eliminarenfermo(p\_inscripcion ENFERMO.INSCRIPCION%TYPE);**

**end pk\_delete;**

**--body**

**create or replace package body pk\_delete**

**AS**

**procedure eliminarempleado(p\_empno EMP.EMP\_NO%TYPE)**

**as**

**begin**

**delete from EMP where EMP\_NO=p\_empno;**

**commit;**

**end;**

**procedure eliminardepartamento(p\_deptno DEPT.DEPT\_NO%TYPE)**

**as**

**begin**

**delete from DEPT where DEPT\_NO=p\_deptno;**

**commit;**

**end;**

**procedure eliminardoctor(p\_doctorno DOCTOR.DOCTOR\_NO%TYPE)**

**as**

**begin**

**delete from DOCTOR where DOCTOR\_NO=p\_doctorno;**

**commit;**

**end;**

**procedure eliminarenfermo(p\_inscripcion ENFERMO.INSCRIPCION%TYPE)**

**as**

**begin**

**delete from ENFERMO where INSCRIPCION=p\_inscripcion;**

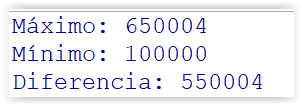
**commit;**

**end;**

**end pk\_delete;**

Si estamos dentro del paquete, podemos utilizar cualquier característica sin necesidad de poner el nombre del paquete, simplemente el nombre del procedimiento o la función que deseemos.

Vamos a crear un procedimiento que tenga tres funciones, una para devolver el máximo salario de los empleados, otra para devolver el mínimo salario y otra para devolver la diferencia entre el máximo y el mínimo.



**CODIGO SQL**

**create or replace package pk\_empleados\_salarios**

**AS**

**function minimo return *number*;**

**function maximo return *number*;**

**function diferencia return *number*;**

**end pk\_empleados\_salarios;**

**--body**

**create or replace package body pk\_empleados\_salarios**

**AS**

**function minimo return *number***

**as**

**v\_minimo EMP.SALARIO%TYPE;**

**begin**

**select min(SALARIO) into v\_minimo from EMP;**

**return v\_minimo;**

**end;**

**function maximo return *number***

**as**

**v\_maximo *number*;**

**begin**

**select max(SALARIO) into v\_maximo from EMP;**

**return v\_maximo;**

**end;**

**function diferencia return *number***

**as**

**v\_diferencia *number*;**

**begin**

**v\_diferencia := maximo - minimo;**

**return v\_diferencia;**

**end;**

**end pk\_empleados\_salarios;**

**select PK\_EMPLEADOS\_SALARIOS.MAXIMO as MAXIMO,**

**PK\_EMPLEADOS\_SALARIOS.MINIMO as MINIMO,**

**PK\_EMPLEADOS\_SALARIOS.DIFERENCIA as DIFERENCIA from DUAL;**

Ejemplo de paquete con las consultas de Update, Delete e Insert sobre la tabla Departamentos (DEPT).

**create or replace package pk\_departamentos**

**as**

**procedure updatedept(p\_id DEPT.DEPT\_NO%TYPE, p\_nombre DEPT.DNOMBRE%TYPE**

**, p\_localidad DEPT.LOC%TYPE);**

**procedure insertdept(p\_id DEPT.DEPT\_NO%TYPE, p\_nombre DEPT.DNOMBRE%TYPE**

**, p\_localidad DEPT.LOC%TYPE);**

**procedure deletedept(p\_id DEPT.DEPT\_NO%TYPE);**

**end pk\_departamentos;**

**--body**

**create or replace package body pk\_departamentos**

**as**

**procedure updatedept(p\_id DEPT.DEPT\_NO%TYPE, p\_nombre DEPT.DNOMBRE%TYPE**

**, p\_localidad DEPT.LOC%TYPE)**

**as**

**begin**

**update DEPT set DNOMBRE=p\_nombre, LOC=p\_localidad**

**where DEPT\_NO=p\_id;**

**commit;**

**end;**

**procedure insertdept(p\_id DEPT.DEPT\_NO%TYPE, p\_nombre DEPT.DNOMBRE%TYPE**

**, p\_localidad DEPT.LOC%TYPE)**

**as**

**begin**

**insert into DEPT values (p\_id, p\_nombre, p\_localidad);**

**commit;**

**end;**

**procedure deletedept(p\_id DEPT.DEPT\_NO%TYPE)**

**as**

**begin**

**delete from DEPT where DEPT\_NO = p\_id;**

**commit;**

**end;**

**end pk\_departamentos;**

**--probando**

**begin**

**--PK\_DEPARTAMENTOS.INSERTDEPT(88, 'martes', 'Madrid');**

**--PK\_DEPARTAMENTOS.UPDATEDEPT(88, 'martesssss', 'MadriZZ');**

**PK\_DEPARTAMENTOS.DELETEDEPT(88);**

**end;**

Necesito una funcionalidad que nos devuelva

el Apellido, el oficio, salario y lugar de trabajo (departamento/hospital)

de todas las personas de nuestra bbdd.

Necesito otra funcionalidad que nos devuelva

el Apellido, el trabajo, salario y lugar de trabajo (departamento/hospital)

dependiendo del salario.

--1) CONSULTA GORDA

**select EMP.APELLIDO, EMP.OFICIO, EMP.SALARIO**

**, DEPT.DNOMBRE**

**from EMP**

**inner join DEPT**

**on EMP.DEPT\_NO=DEPT.DEPT\_NO**

**UNION**

**select DOCTOR.APELLIDO, DOCTOR.ESPECIALIDAD, DOCTOR.SALARIO**

**, HOSPITAL.NOMBRE**

**from DOCTOR**

**inner join HOSPITAL**

**on DOCTOR.HOSPITAL\_COD=HOSPITAL.HOSPITAL\_COD**

**UNION**

**select PLANTILLA.APELLIDO, PLANTILLA.FUNCION, PLANTILLA.SALARIO**

**, HOSPITAL.NOMBRE**

**from PLANTILLA**

**inner join HOSPITAL**

**on PLANTILLA.HOSPITAL\_COD=HOSPITAL.HOSPITAL\_COD;**

--2) CONSULTA DENTRO DE VISTA

**create or replace view V\_TODOS\_EMPLEADOS as**

**select EMP.APELLIDO, EMP.OFICIO, EMP.SALARIO**

**, DEPT.DNOMBRE**

**from EMP**

**inner join DEPT**

**on EMP.DEPT\_NO=DEPT.DEPT\_NO**

**UNION**

**select DOCTOR.APELLIDO, DOCTOR.ESPECIALIDAD, DOCTOR.SALARIO**

**, HOSPITAL.NOMBRE**

**from DOCTOR**

**inner join HOSPITAL**

**on DOCTOR.HOSPITAL\_COD=HOSPITAL.HOSPITAL\_COD**

**UNION**

**select PLANTILLA.APELLIDO, PLANTILLA.FUNCION, PLANTILLA.SALARIO**

**, HOSPITAL.NOMBRE**

**from PLANTILLA**

**inner join HOSPITAL**

**on PLANTILLA.HOSPITAL\_COD=HOSPITAL.HOSPITAL\_COD;**

**select \* from V\_TODOS\_EMPLEADOS;**

--3) PAQUETE CON DOS PROCEDIMIENTOS

**create or replace package pk\_vista\_empleados**

**as**

**procedure todos\_empleados;**

**procedure todos\_empleados\_salario(p\_salario EMP.SALARIO%TYPE);**

**end pk\_vista\_empleados;**

**--body**

**create or replace package body pk\_vista\_empleados**

**as**

**procedure todos\_empleados**

**as**

**cursor c\_empleados is**

**select \* from V\_TODOS\_EMPLEADOS;**

**begin**

**for v\_emp in c\_empleados**

**loop**

**dbms\_output.put\_line(v\_emp.APELLIDO || ', Oficio: '**

**|| v\_emp.OFICIO || ', Salario: ' || v\_emp.SALARIO**

**|| ', Lugar: ' || v\_emp.DNOMBRE);**

**end loop;**

**end;**

**procedure todos\_empleados\_salario(p\_salario EMP.SALARIO%TYPE)**

**as**

**cursor c\_empleados is**

**select \* from V\_TODOS\_EMPLEADOS**

**where SALARIO >= p\_salario;**

**begin**

**for v\_emp in c\_empleados**

**loop**

**dbms\_output.put\_line(v\_emp.APELLIDO || ', Oficio: '**

**|| v\_emp.OFICIO || ', Salario: ' || v\_emp.SALARIO**

**|| ', Lugar: ' || v\_emp.DNOMBRE);**

**end loop;**

**end;**

**end pk\_vista\_empleados;**

--3A) PROCEDIMIENTO PARA DEVOLVER TODOS LOS DATOS EN UN CURSOR

--3B) PROCEDIMIENTO PARA DEVOLVER TODOS LOS DATOS EN UN CURSOR POR SALARIO

**begin**

**--PK\_VISTA\_EMPLEADOS.TODOS\_EMPLEADOS;**

**PK\_VISTA\_EMPLEADOS.TODOS\_EMPLEADOS\_SALARIO(350000);**

**end;**

Necesitamos un paquete con procedimiento para modificar el salario de cada

Doctor de forma individual.

La modificación de los datos de cada doctor será de forma aleatoria.

Debemos comprobar el Salario de cada Doctor para ajustar el número aleatorio

del incremento.

1) Doctor con menos de 200.000: Incremento aleatorio de 500

2) Doctor entre de 200.000 y 300.000: Incremento aleatorio de 300

3) Doctor mayor a 300.000: Incremento aleatorio de 50

El incremento Random lo haremos con una función dentro del paquete.

**create or replace package pk\_doctores**

**as**

**procedure incremento\_random\_doctores;**

**function function\_random\_doctores(p\_iddoctor DOCTOR.DOCTOR\_NO%TYPE)**

**return *NUMBER*;**

**end pk\_doctores;**

**--body**

**create or replace package body pk\_doctores**

**as**

**procedure incremento\_random\_doctores**

**as**

**cursor c\_doctores is**

**select DOCTOR\_NO, APELLIDO, SALARIO from DOCTOR;**

**v\_random *number*;**

**begin**

**for v\_doc in c\_doctores**

**loop**

**v\_random := function\_random\_doctores(v\_doc.DOCTOR\_NO);**

**update DOCTOR set SALARIO = SALARIO + v\_random**

**where DOCTOR\_NO=v\_doc.DOCTOR\_NO;**

**dbms\_output.put\_line('Doctor ' || v\_doc.APELLIDO**

**|| ' tiene un incremento de ' || v\_random);**

**end loop;**

**end;**

**function function\_random\_doctores(p\_iddoctor DOCTOR.DOCTOR\_NO%TYPE)**

**return *NUMBER***

**as**

**v\_salario DOCTOR.SALARIO%TYPE;**

**v\_random *NUMBER*;**

**begin**

**select SALARIO into v\_salario from DOCTOR**

**where DOCTOR\_NO=p\_iddoctor;**

**if (v\_salario < 200000) then**

**v\_random := trunc(dbms\_random.value(1,500));**

**elsif (v\_salario > 300000) then**

**v\_random := trunc(dbms\_random.value(1, 50));**

**else**

**v\_random := trunc(dbms\_random.value(1, 300));**

**end if;**

**return v\_random;**

**end;**

**end pk\_doctores;**

**declare**

**cursor c\_doctores is**

**select DOCTOR\_NO, APELLIDO, SALARIO from DOCTOR;**

**v\_random *number*;**

**begin**

**for v\_doc in c\_doctores**

**loop**

**v\_random := random\_doctor(v\_doc.DOCTOR\_NO);**

**update DOCTOR set SALARIO = SALARIO + v\_random**

**where DOCTOR\_NO=v\_doc.DOCTOR\_NO;**

**dbms\_output.put\_line('Doctor ' || v\_doc.APELLIDO**

**|| ' tiene un incremento de ' || v\_random);**

**end loop;**

**end;**

**create or replace function random\_doctor**

**(p\_iddoctor DOCTOR.DOCTOR\_NO%TYPE)**

**return *number***

**as**

**v\_salario DOCTOR.SALARIO%TYPE;**

**v\_random *NUMBER*;**

**begin**

**select SALARIO into v\_salario from DOCTOR**

**where DOCTOR\_NO=p\_iddoctor;**

**if (v\_salario < 200000) then**

**v\_random := trunc(dbms\_random.value(1,500));**

**elsif (v\_salario > 300000) then**

**v\_random := trunc(dbms\_random.value(1, 50));**

**else**

**v\_random := trunc(dbms\_random.value(1, 300));**

**end if;**

**return v\_random;**

**end;**

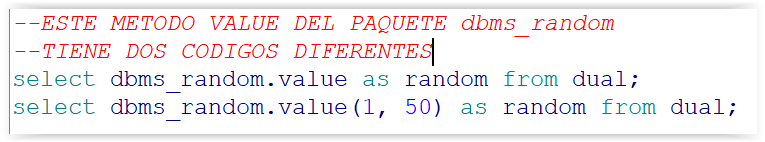
**SOBRECARGA DE PROCEDIMIENTOS O FUNCIONES**

Un paquete también permite la sobrecargar dentro de la lógica de un procedimiento o una función.

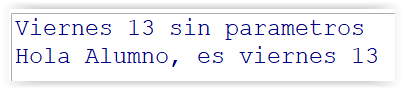
Una sobrecarga indica que, con un mismo nombre de procedimiento o función, tiene varias opciones para enviar parámetros y realizará diferentes códigos.

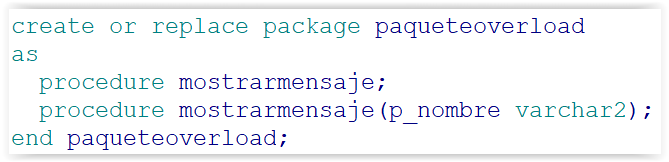
Es parecido a parámetros por defecto, con la diferencia de que tenemos varios procedimientos o funciones.

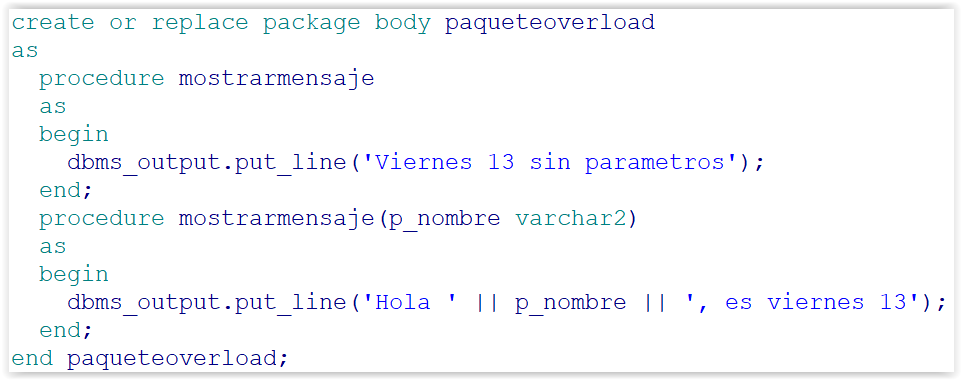
Dependiendo del número de parámetros que reciba un procedimiento o función, realizará un código u otro.

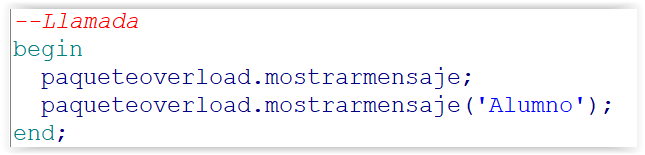


Ejemplo de un paquete en Oracle con sobrecarga en procedimientos.









**CODIGO SQL**

create or replace package body paqueteoverload

as

procedure mostrarmensaje

as

begin

dbms\_output.put\_line('Viernes 13 sin parametros');

end;

procedure mostrarmensaje(p\_nombre varchar2)

as

begin

dbms\_output.put\_line('Hola ' || p\_nombre || ', es viernes 13');

end;

end paqueteoverload;

--Llamada

begin

paqueteoverload.mostrarmensaje;

paqueteoverload.mostrarmensaje('Alumno');

end;

**EJEMPLO PAQUETE CON SOBRECARGA PARA MODIFICAR DEPARTAMENTOS**

*--CREAR UN PAQUETE PARA MODIFICAR DEPARTAMENTOS*

*--TENDREMOS UN PROCEDIMIENTO QUE MODIFICARA TODOS LOS DATOS*

*--DEL DEPARTAMENTO. TODOS MENOS EL ID.*

*--OTRO PROCEDIMIENTO PARA MODIFICAR SOLO LA LOCALIDAD.*

*--CABECERA DEL PAQUETE*

create or replace package paqueteupdatedept

as

procedure modificardepartamento

(p\_num dept.dept\_no%type,

p\_nom dept.dnombre%type,

p\_loc dept.loc%type);

procedure modificardepartamento

(p\_num dept.dept\_no%type,

p\_loc dept.loc%type);

end paqueteupdatedept;

create or replace package body paqueteupdatedept

as

procedure modificardepartamento

(p\_num dept.dept\_no%type,

p\_nom dept.dnombre%type,

p\_loc dept.loc%type)

as

begin

update dept set dnombre = p\_nom, loc = p\_loc

where dept\_no = p\_num;

end;

procedure modificardepartamento

(p\_num dept.dept\_no%type,

p\_loc dept.loc%type)

as

begin

update dept set loc = p\_loc

where dept\_no = p\_num;

end;

end paqueteupdatedept;

select \* from dept;

*--llamada*

begin

*--paqueteupdatedept.modificardepartamento(40, 'CONTABLES', 'ALAVA');*

paqueteupdatedept.modificardepartamento(40, 'CADIZ');

end;

**EJEMPLO PAQUETE CON TODAS LAS ACCIONES CRUD SOBRE DEPARTAMENTOS**

**CRUD: CREATE, READ, UPDATE Y DELETE**

*--Realizar un paquete que contenga las acciones típicas sobre departamentos:*

*-- INSERTAR, MODIFICAR, ELIMINAR*

*-- MOSTRAR TODOS LOS DEPARTAMENTOS*

create or replace package paquetedepartamentos

as

procedure insertar *--insert into dept values (3 campos)*

(p\_num dept.dept\_no%type

, p\_nom dept.dnombre%type

, p\_loc dept.loc%type);

procedure modificar *--update dept set dnombre=campo, loc = campo where campo*

(p\_num dept.dept\_no%type

, p\_nom dept.dnombre%type

, p\_loc dept.loc%type);

procedure eliminar *--delete from dept where numero = campo*

(p\_num dept.dept\_no%type);

procedure mostrar;

end paquetedepartamentos;

*--CUERPO DEL PAQUETE*

create or replace package BODY paquetedepartamentos

as

procedure insertar

(p\_num dept.dept\_no%type

, p\_nom dept.dnombre%type

, p\_loc dept.loc%type)

as

begin

insert into dept values (p\_num, p\_nom, p\_loc);

end;

procedure modificar

(p\_num dept.dept\_no%type

, p\_nom dept.dnombre%type

, p\_loc dept.loc%type)

as

begin

update dept set dnombre = p\_nom, loc = p\_loc

where dept\_no = p\_num;

end;

procedure eliminar

(p\_num dept.dept\_no%type)

as

begin

delete from dept where dept\_no = p\_num;

end;

procedure mostrar

as

cursor c\_dept is

select \* from dept;

begin

for dept in c\_dept

loop

dbms\_output.put\_line(dept.dnombre || ', ' || dept.loc);

end loop;

end;

end paquetedepartamentos;

select \* from dept;

*--Llamada*

begin

*--paquetedepartamentos.insertar(50, 'I+D', 'EIBAR');*

*--paquetedepartamentos.modificar(50, 'INFORMATICA', 'BURGOS');*

*--paquetedepartamentos.eliminar(50);*

paquetedepartamentos.mostrar;

end;