

# PROCEDIMIENTOS Y FUNCIONES EN UNA BASE DE DATOS

Trabajo de Alejandro Sainz Sainz

BD-  
ACTIVIDAD 4.4

Alejandro Sainz Sainz

COMIENZO .....	3
EJERCICIO 1 .....	4
EJERCICIO 2 .....	5
EJERCICIO 3 .....	6
EJERCICIO 4 .....	8
EJERCICIO 5 .....	9
EJERCICIO 6 .....	10
EJERCICIO 7 .....	12
EJERCICIO 8 .....	14

Alejandro Sainz Sainz

1 PRIMER PROCEDIMIENTO .....	4
2 LLAMADA AL PROCEDIMIENTO.....	4
3 SELECT @RESULTADO .....	4
4 CREACION DEL PROCEDIMIENTO.....	5
5 DEFINIMOS UNA VARIABLE EXTERNA .....	5
6 LLAMADA AL PROCEDIMIENTO.....	5
7 RESULTADO DEL PROCEDIMIENTO .....	6
8 RESULTADO.....	6
9 CODIGO DE LA FUNCION .....	6
10 EJECUCIÓN DE LA FUNCION.....	7
11 RESULTADO DE LA FUNCIÓN .....	7
12 CODIGO DEL PROCEDIMIENTO .....	8
13 EJECUCIÓN DEL PROCEDIMIENTO.....	8
14 OBTENEMOS EL RESULTADO .....	8
15 RESULTADO.....	8
16 CODIGO DE LA FUNCIÓN .....	9
17 SELECT DIA_SEMANA(3).....	9
18 CODIGO DE LA FUNCION .....	10
19 FUNCIONES A EJECUTAR .....	10
20 CODIGO DEL PROCEDIMIENTO .....	12
21 LLAMAMOS AL PROCEDIMIENTO.....	13
22 SELECT @VALORFACTORIAL.....	13
23 LAS TABLAS .....	14
24 CODIGO .....	15
25 LLAMADA A LA FUNCION.....	15
26 OBTENEMOS LAS TABLAS .....	15
27 RESULTADO.....	16

## COMIENZO

---

En este ejercicio vamos a resolver una serie de ejercicios con procedimientos y funciones. Estos elementos nos ayudan a mantener una BD mediante una serie de procedimientos automatizados que harán que no tengamos que realizar manualmente operaciones repetitivas.

A diferencia de los triggers que hemos usado con anterioridad, los procedimientos y funciones pueden recibir parámetros como argumentos para luego devolver el resultado esperado.

## EJERCICIO 1

**Primero de primaria.** escribe un procedimiento que devuelva como resultado la suma de dos números enteros los cuales se le pasan como parámetros.

```
7 • drop procedure if exists suma_numeros;
8
9   delimiter //
0 • create procedure suma_numeros(in num1 int, in num2 int, out resultado int)
1   begin
2       set resultado = num1 + num2;
3   end//
```

### 1 PRIMER PROCEDIMIENTO

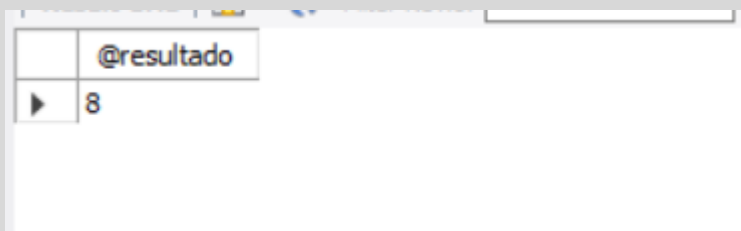
Lo primero es crear el procedimiento. En este caso le llamo suma\_numero, y lo que hago es pasarle 3 parámetros, dos como in y uno como out, los tres del tipo into.

Dentro del procedimiento lo que hago es hacer que el valor de la variable out resultado sea igual a la suma de los otros dos números.

```
call suma_numeros(3,5,@resultado);
```

### 2 LLAMADA AL PROCEDIMIENTO

Con este código llamo al procedimiento y le paso dos números como parámetros, que son los que se van a sumar y le indico que como tercer parámetro, el parámetro out, va a ser una variable llamada resultado.



	@resultado
▶	8

### 3 SELECT @RESULTADO

Con el comando mostrado como nombre de la imagen anterior, la consola de workbench nos devuelve el siguiente resultado.

## EJERCICIO 2

**¿Segundo de primaria?.** Escribe un procedimiento que devuelva como resultados la suma y la multiplicación de dos números enteros. El resultado de la multiplicación debe devolverse en la misma variable que determina el segundo número.

```
17 |
18 drop procedure if exists suma_producto;
19
20 delimiter //
21 • create procedure suma_producto(in num1 int, inout num2 int, out res_suma int)
22 begin
23     set res_suma = num1 + num2;
24     set num2 = num1 * num2;
25 end//
26
```

### 4 CREACION DEL PROCEDIMIENTO

Este es el código del segundo procedimiento. Lo llamo suma\_producto. En este caso paso tres parámetros, uno como in, otro como inout y un último sólo como out. Como dice el ejercicio el segundo parámetro para un valor al procedimiento y además de eso almacena un resultado.

Como se explico en clase, el código que almacena la multiplicación se debe colocar el último para que no se modifique el resultado del segundo parámetro antes de tiempo y nos de un resultado que no sea el esperado.

```
set @num2 = 5;
```

### 5 DEFINIMOS UNA VARIABLE EXTERNA

Uno de los motivos por los que me daba error en clase era porque intentaba definir la variable y el valor como parámetro, y no obtenía ningún tipo de resultado favorable, ni de ningún otro tipo, por lo que se explicó que debíamos de crear una variable fuera del procedimiento, y luego pasar esa variable como parámetro. Al hacer esto, el valor almacenado en la variable era el que se usaba como in y la propia variable era el contenedor que se usaría como out.

```
call suma_producto(3, @num2, @res_suma);
```

### 6 LLAMADA AL PROCEDIMIENTO

Lo que vemos en la imagen es un resumen de lo que he explicado anteriormente.

```

0      select @num2,@res_suma;
1

```

#### 7 RESULTADO DEL PROCEDIMIENTO

Una vez llamado el procedimiento lo que debemos de hacer es un select a las variables para que se nos devuelva su resultado.

	@num2	@res_suma
▶	15	8

#### 8 RESULTADO

Después de ejecutar el select este es el resultado que nos devuelve la ejecución del procedimiento.

## EJERCICIO 3

**Recuperación del primer ciclo.** Rehaz la actividad 1 con una función en lugar de un procedimiento.

```

1
2      drop function if exists recuperacion;
3
4      delimiter //
5 •   create function recuperacion(numero1 int, numero2 int)
6       returns int
7       deterministic
8       begin
9           return numero1 + numero2;
0       end //
1

```

#### 9 CODIGO DE LA FUNCION

Bien, en este caso creamos la función, que ejecuta la misma operación que el procedimiento del ejercicio 1. Como diferencia indicar simplemente los parámetros y de que tipo son, que devolverá un int y que es una función determinista.

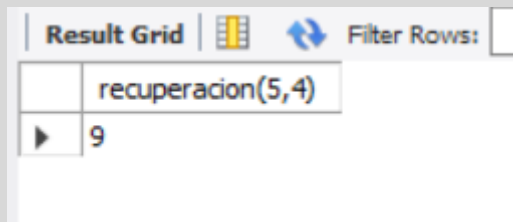
Después la operación que llevará a cabo es la misma que el ejercicio 1.

Alejandro Sainz Sainz

```
select recuperacion(5,4);
```

**10 EJECUCIÓN DE LA FUNCION**

Ejecutamos la función con el comando select e indicamos los parámetros que nos interesan, dos números del tipo entero.



The screenshot shows a 'Result Grid' window with a 'Filter Rows' button. The grid contains one row with the header 'recuperacion(5,4)' and the value '9'.

	recuperacion(5,4)
▶	9

**11 RESULTADO DE LA FUNCIÓN**

Aquí vemos el resultado que nos devuelven la función después de ejecutarla.



## EJERCICIO 4

**Día sin IVA.** Crea un procedimiento que reciba como parámetro un precio y calcule su precio sin IVA (considera IVA al 21%), devolviéndolo en una variable.

```
drop procedure if exists iva;

delimiter //
• create procedure iva(in precio decimal(6,2),out precio_sin_iva decimal(6,2))
begin
    set precio_sin_iva = (precio/1.21);
end //
```

### 12 CODIGO DEL PROCEDIMIENTO

Creamos este procedimiento y, creo que está bien así, insertamos el código que necesitamos para que nos calcule el precio de un producto sin el iva. Como parámetros pasamos un decimal como in y otro como out.

```
call iva (1000.00,@precio_sin_iva);
```

### 13 EJECUCIÓN DEL PROCEDIMIENTO

Ejecutamos el procedimiento y le pasamos dos parámetros, un valor y una variable que almacenará el resultado. Hay que comentar que al ejecutar me salta un warning, pero luego se ejecuta sin problemas. No estoy seguro de si será por el tipo de datos o por la forma en la que he ejecutado la operación de cálculo.

```
select @precio_sin_iva;
```

### 14 OBTENEMOS EL RESULTADO

Comprobamos cual es el resultado.

@precio_sin_iva
▶ 826.45

### 15 RESULTADO

Nos devuelve el resultado sin problema. No se por qué saltará el warning.

## EJERCICIO 5

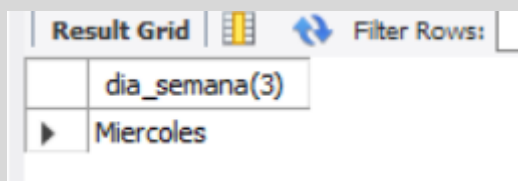
**Del calendario no.** Crear una función para mostrar el día de la semana según el valor de entrada numérico: 1 para lunes, 2 para martes, etc...

```
55 drop function if exists dia_semana;  
56  
57 delimiter //  
58 • create function dia_semana(numero int)  
59 returns enum('Lunes','Martes','Miercoles','Jueves','Viernes','Sabado','Domingo')  
60 deterministic  
61 begin  
62     return case  
63         when numero = 1 then 'Lunes'  
64         when numero = 2 then 'Martes'  
65         when numero = 3 then 'Miercoles'  
66         when numero = 4 then 'Jueves'  
67         when numero = 5 then 'Viernes'  
68         when numero = 6 then 'Sabado'  
69         when numero = 7 then 'Domingo'  
70     end;  
71 end //
```

### 16 CODIGO DE LA FUNCIÓN

Este es el código que he generado para resolver esta función. No sé si habrá una forma mejor de hacerlo o más corta, pero es la que he usado yo.

Indico un número como parámetro y retorno un enum con el nombre de los días de la semana. Con un return case devolvemos un día u otro según el valor del número indicado.



### 17 SELECT DIA\_SEMANA(3)

Con el comando que vemos como título de la foto obtenemos el siguiente resultado. Si usamos un número mayor que 7 nos devuelve null. Lo he comprobado simplemente para saber si ocurría alguna otra cosa.

## EJERCICIO 6

**Esto compila... y calcula.** Crear una función calculadora que realice operaciones con dos números decimales. La operación a realizar depende de un tercer parámetro que puede ser suma, resta, mult o div.

```
75 drop function if exists calculadora;  
76  
77 delimiter //  
78 • create function calculadora(numero1 decimal(5,2), numero2 decimal(5,2), operacion int)  
79 returns decimal(10,2)  
80 deterministic  
81 begin  
82     return case  
83         when operacion = 1 then numero1 + numero2  
84         when operacion = 2 then numero1 - numero2  
85         when operacion = 3 then numero1 * numero2  
86         when operacion = 4 then numero1/numero2  
87     end;  
88 end //  
89
```

### 18 CODIGO DE LA FUNCION

Este es el código que he generado. En este caso se reciben como parámetros un int, que indicará el tipo de operación, y dos decimales que serán los números con los que vamos a trabajar. Devolverá un decimal y es una función determinista.

Mediante un return case evaluaré el int para saber cual es el tipo de operación y según ese valor devolveremos un resultado u otro.

```
89  
90 select calculadora(100.99,90.99,1);  
91 select calculadora(100.99,90.99,2);  
92 select calculadora(100.99,90.99,3);  
93 select calculadora(100.99,90.99,4);  
94
```

### 19 FUNCIONES A EJECUTAR

Estas son las funciones que voy a ejecutar, una por cada tipo de operación.

Result Grid		Filter Rows:
	calculadora(100.99,90.99,1)	
▶	191.98	

Result Grid		Filter Rows:
	calculadora(100.99,90.99,2)	
▶	10.00	

	calculadora(100.99,90.99,3)	
▶	9189.08	

Result Grid		Filter Rows:
	calculadora(100.99,90.99,4)	
▶	1.11	

Estos son los diferentes resultados que obtenemos según la operación que hemos ido realizando.

## EJERCICIO 7

**El factorial.** crea un procedimiento que calcule el factorial de N. N será un número proporcionado por el usuario como argumento al procedimiento

```
delimiter //
• create procedure factorial(in n int, out valorFactorial int)
begin
    declare nInicial int;
    set nInicial = 2;
    set valorFactorial = 1;

    bucle : loop
        set valorFactorial=valorFactorial * nInicial;
        set nInicial = nInicial + 1;
        if nInicial > n then
            leave bucle;
        end if;
    end loop bucle;
end //
```

### 20 CODIGO DEL PROCEDIMIENTO

Este es el código que he generado para este procedimiento. Recibe dos parámetros, un in que será un int para el número límite del factorial que vamos a calcular, y otro int como out para el resultado.

En este caso calculo el valor inicial, que será dos, por que el 1 podemos obviarle a la hora de hacer el cálculo, aunque se puede incluir sin ningún tipo de problema. Luego definimos el valor del factorial que también comienza como 1, ya que si fuese cero el factorial siempre sería cero.

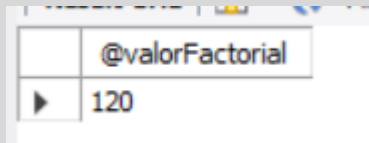
Luego definimos un loop, con el nombre que queramos, y vamos multiplicando el valor del factorial por el valor actual del número inicial. Incrementamos después de esto el número inicial en 1 y evaluamos. Si el número es mayor que el número introducido por parámetro debe detenerse y salir del loop.

Alejandro Sainz Sainz

```
call factorial(5,@valorFactorial);
```

**21 LLAMAMOS AL PROCEDIMIENTO**

Al llamar al procedimiento le indicamos como parámetro cual es el número del que queremos calcular el factorial.



	@valorFactorial
▶	120

**22 SELECT @VALORFACTORIAL**

Con el comando que vemos acompañando a la imagen obtenemos el resultado de el procedimiento ejecutado, en este caso, 120.

## EJERCICIO 8

**El mismo saco.** Crea un procedimiento que introduce en una tabla denominada "impares" los primeros 50 números impares.

```
18
19 create table resultado(
20     id int auto_increment not null primary key,
21     resultado int not null
22 );
23
24 create temporary table resultado2(
25     id int auto_increment not null primary key,
26     resultado int not null
27 );
28
```

### 23 LAS TABLAS

Para resolver este ejercicio tenemos que crear una tabla. Yo aquí creo dos, pues se podría hacer con una tabla normal o con una temporal, que se borraría al cerrar la sesión pero que nos puede servir de la misma forma para almacenar los resultados.

```

31 delimiter //
32 • create procedure impares()
33 begin
34     declare contador int;
35     set contador = 1;
36
37     bucle_impares : loop
38
39         if mod(contador,2)<>0 then
40             insert into resultado (resultado) values (contador);
41             insert into resultado2 (resultado) values (contador);
42         end if;
43         set contador = contador + 1;
44         if contador > 50 then
45             leave bucle_impares;
46         end if;
47     end loop bucle_impares;
48 end //

```

**24 CODIGO**

Con el código que vemos arriba resuelvo el ejercicio.

Definimos un contador y le damos como valor 1.

Abrimos un loop, al que llamamos como más nos convenga. Después de esto, con la función MOD, evaluamos que al dividir el contador entre dos sea distinto de cero.

De cumplirse esa condición insertamos ese valor en las dos tablas.

Luego incrementamos en 1 el valor del contador y evaluamos si ese es mayor de 50. En caso de serlo se cierra el bucle.

```

• call impares();

```

**25 LLAMADA A LA FUNCION**

Ahora lo que tenemos que hacer es llamar a la función y ver si se han insertado los datos en cualquiera de las dos tablas.

```

51 select * from resultado;
52 select * from resultado2;

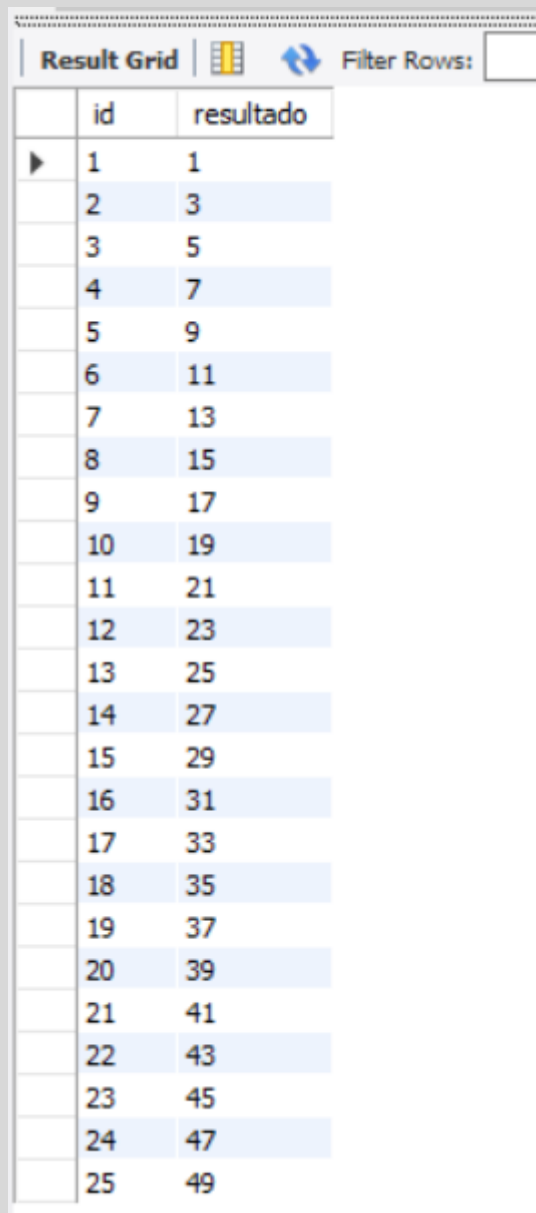
```

**26 OBTENEMOS LAS TABLAS**



Alejandro Sainz Sainz

Con los comandos que vemos en la imagen anterior obtenemos las dos tablas.



	id	resultado
▶	1	1
	2	3
	3	5
	4	7
	5	9
	6	11
	7	13
	8	15
	9	17
	10	19
	11	21
	12	23
	13	25
	14	27
	15	29
	16	31
	17	33
	18	35
	19	37
	20	39
	21	41
	22	43
	23	45
	24	47
	25	49

**27 RESULTADO**

Aquí vemos como queda la tabla. Como es muy larga no voy a poner una imagen de la temporal que es exactamente igual.