

Procedimientos almacenados y funciones - Introducción

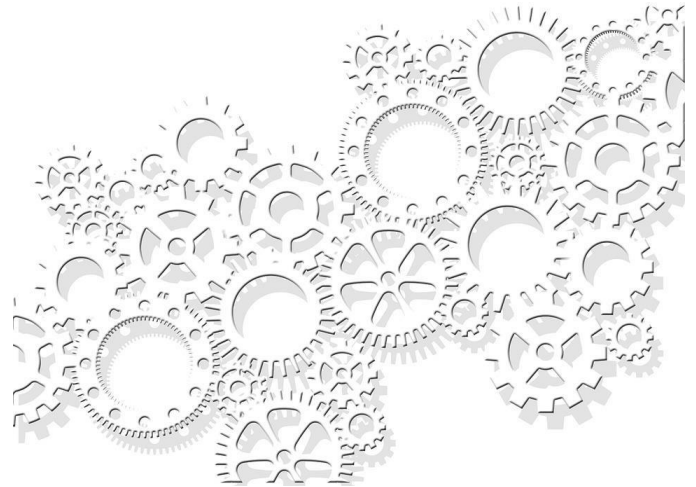


Introducción

Es un conjunto de comandos o instrucciones SQL que realizan una función específica y que se pueden utilizar tanto en consultas como en scripts.

Hay dos tipos:

- Procedimientos almacenados
- Funciones



Características

Algunas de las características son las siguientes:

- Pueden recibir y devolver parámetros.
- Pueden manejar tablas, ejecutando operaciones e iteraciones de lectura/escritura.
- Se almacenan en la base de datos en la cual se crean.
- No dependen de ninguna tabla en particular.
- Pueden aceptar recursividad.

Ventajas

Algunas de las ventajas son las siguientes:

- **Reutilización del código:** las funciones implementadas se pueden utilizar desde cualquier sitio de la base de datos o de la aplicación.
- **Seguridad:** se puede limitar el acceso a usuarios.
- **Centralización:** mantenimiento más fácil para determinadas funciones que pueden ser complejas.
- **Sencillez:** las funciones pueden implementar funcionalidades complejas que los usuarios pueden utilizar directamente.

Procedimientos almacenados VS funciones

Una función y un procedimiento almacenado tienen muchas similitudes, pero no son iguales.

Procedimientos almacenados:

- Puede tener parámetros de entrada y salida.
- Puede retornar ningún, uno o varios parámetros.
- No se pueden utilizar dentro de consultas (SELECT).

Funciones:

- Solo puede tener parámetros de entrada.
- Siempre retorna un solo valor.
- Se pueden utilizar dentro de consultas (SELECT).

Sintaxis Procedimientos Almacenados

La sintaxis para crear un procedimiento almacenado es la siguiente:

```
CREATE PROCEDURE nombre_procedimiento (parametros)
```

```
BEGIN
```

```
.....
```

```
END
```

Parámetros: Se pueden incluir uno, varios o ningún parámetro separado por comas de la siguiente manera:

```
IN | OUT | INOUT nombre_parametro TIPO_PARAMETRO
```

Procedimientos: Crear y ejecutar

```
118 delimiter //
119 • CREATE PROCEDURE muestra_libros()
120 BEGIN
121     select * from libros;
122 END //
123 delimiter ;
124
125 • call muestra_libros;
126
127
```

Establecemos el delimitador de línea a `//` (doble barra) para que no se ejecuten las líneas SQL incluidas en el procedimiento almacenado. Al finalizar la definición del procedimiento, volvemos a establecer el delimitador de línea a `;` (punto y coma).

Para ejecutar los procedimientos se utiliza la cláusula `CALL` más el nombre del procedimiento.

En este ejemplo se realiza una consulta de los libros existentes en la tabla.

result Grid			Filter Rows:	Export:
id	titulo	autor		
1	El Quijote	Cervantes		
6	Juego de tronos	George R. R. Martin		
88	El cazador de sueños	Stephen King		
99	Hamlet	William Shakespeare		
123	Poeta en Nueva York	Federico García Lorca		

Procedimientos: Parámetros de entrada

Creamos un procedimiento que reciba un parámetro de entrada y lo utilice para filtrar la consulta incluida en él.

```
118 delimiter //
119 • CREATE PROCEDURE libros_por_autor(in parametro_autor varchar(50))
120 BEGIN
121     select * from libros where autor=parametro_autor;
122 END //
123 delimiter ;
124
125 • call libros_por_autor('Cervantes');
126
127
```

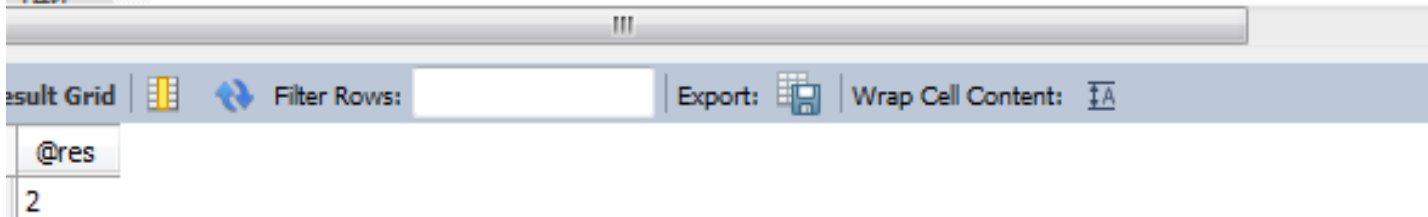
Result Grid			Filter Rows:	Export:	Wrap Cell Content:
id	titulo	autor			
1	El Quijote	Cervantes			

Procedimientos: Parámetros de salida

Creamos un procedimiento que reciba dos parámetros de entrada y uno de salida.

```
130 delimiter //
131 • CREATE PROCEDURE suma(in parametro1 int,in parametro2 int,out resultado int)
132   BEGIN
133     select parametro1+parametro2 into resultado;
134   END //
135 delimiter ;
136
137 • call suma(1,1,@res);
138
139 • select @res;
```

Introducimos el resultado en una variable llamada res.



Sintaxis Funciones

La sintaxis para crear una función es la siguiente:

```
CREATE FUNCTION nombre_funcion (parametros) RETURNS TIPO_PARAMETRO
```

```
BEGIN
```

```
.....
```

```
RETURN ....
```

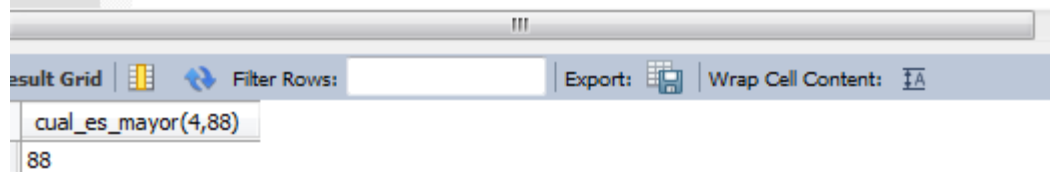
```
END
```

Ejecutar la función: se puede ejecutar la función invocando su nombre desde, por ejemplo, una sentencia SELECT: *SELECT nombre_funcion();*

Funciones: Ejemplo con parámetros

Creamos una función que reciba dos parámetros enteros y nos devuelve el valor del mayor de los dos.

```
143 delimiter //
144 • create function cual_es_mayor(valor1 int,valor2 int) returns int
145 begin
146     if valor1>valor2 then
147         return valor1;
148     else
149         return valor2;
150     end if;
151 end //
152 delimiter ;
153
154 • select cual_es_mayor(4,88);
155
```



result Grid
cual_es_mayor(4,88)
88

Eliminar

Para eliminar procedimientos almacenados y funciones se utiliza la sentencia **DROP**:

- Procedimientos:

```
DROP PROCEDURE nombre_procedimiento;
```

- Funciones:

```
DROP FUNCTION nombre_funcion;
```