# Procedimientos almacenados

Los procedimientos almacenados se crean en la base de datos seleccionada, excepto los procedimientos almacenados temporales, que se crean en la base de datos "tempdb".

En primer lugar se deben probar las instrucciones que se incluyen en el procedimiento almacenado. Después, si se obtiene el resultado esperado, se crea el procedimiento.

Los procedimientos almacenados pueden hacer referencia a tablas, vistas, a funciones definidas por el usuario, a otros procedimientos almacenados y a tablas temporales.

Un procedimiento almacenado pueden incluir cualquier cantidad y tipo de instrucciones, excepto:

create default, create procedure, create rule, create trigger y create view. Se pueden crear otros objetos (por ejemplo índices, tablas), en tal caso deben especificar el nombre del propietario; se pueden realizar inserciones, actualizaciones, eliminaciones, etc.

Si un procedimiento almacenado crea una tabla temporal, dicha tabla sólo existe dentro del procedimiento y desaparece al finalizar el mismo. Lo mismo sucede con las variables.

## 1.- Creación de procedimientos almacenados

Primero nos colocaremos en la base de datos en la que queramos crear nuestro procedimiento

Después hacemos clic en Nueva consulta.

Para crear un procedimiento almacenado empleamos la instrucción "create procedure" o “create proc”

La sintaxis básica parcial es:

```
create procedure NOMBREPROCEDIMIENTO
 as INSTRUCCIONES;
```

Ejemplo:

Con las siguientes instrucciones creamos un procedimiento almacenado llamado "pa_libros_limite_stock" que muestra todos los libros de los cuales hay menos de 10 disponibles:

```
create proc pa_libros_limite_stock
  as
   select *from libros
   where cantidad <=10;
```

De esta forma, creamos un procedimiento almacenado colocando "create procedure" (o "create proc", que es la forma abreviada), luego el nombre del procedimiento y seguido de "as" las sentencias que definen el procedimiento.

"create procedure" debe ser la primera sentencia de un lote. Y el procedimiento almacenado terminará con un punto y coma.

Para ejecutar el procedimiento almacenado creado anteriormente tecleamos:

```
exec pa_libros_limite_stock;
```

Así, para ejecutar un procedimiento almacenado colocamos "execute" (o "exec") seguido del nombre del procedimiento.

Vamos a crear un procedimiento que elimine la tabla libros y vuelva a crearla con los datos que nos interesa que contenga. De esta forma, cada vez que queramos que la tabla vuelva a sus valores originales solo deberemos ejecutar el procedmiento:

```
create procedure pa_crear_libros
 as
 if object_id('libros')is not null
  drop table libros;
 create table libros(
  codigo int identity,
  titulo varchar(40),
  autor varchar(30),
  editorial varchar(20),
  precio decimal(5,2),
  primary key(codigo)
 );

 insert into libros values('Uno','Richard Bach','Planeta',15);
 insert into libros values('Ilusiones','Richard Bach','Planeta',18);
 insert into libros values('El aleph','Borges','Emece',25);
 insert into libros values('Aprenda PHP','Mario Molina','Nuevo siglo',45);
```

```
  insert into libros values('Matematica estas ahi','Paenza','Nuevo siglo',12);
  insert into libros values('Java en 10 minutos','Mario Molina','Paidos',35);
```

Después, lo ejecutaremos cada vez que empecemos un nuevo ejercicio tecleando la siguiente instrucción

```
 exec pa_crear_libros;
```

## 2.- Eliminación de procedimientos almacenados

Los procedimientos almacenados se eliminan con "drop procedure". Sintaxis:

```
 drop procedure NOMBREPROCEDIMIENTO;
```

Si el procedimiento que queremos eliminar no existe, aparece un mensaje de error, para evitarlo, podemos emplear esta sintaxis:

```
 if object_id('NOMBREPROCEDIMIENTO') is not null
  drop procedure NOMBREPROCEDIMIENTO;
```

Por ejemplo: eliminamos, si existe, el procedimiento "pa_libros_autor", si no existe, mostramos un mensaje:

```
 if object_id('pa_libros_autor') is not null
  drop procedure pa_libros_autor
 else
  select 'No existe el procedimiento "pa_libros_autor"';
```

"drop procedure" puede abreviarse con "drop proc".

Podemos eliminar una tabla que dependa un procedimiento, SQL Server lo permite, pero luego, al ejecutar el procedimiento, aparecerá un mensaje de error porque la tabla referenciada no existe.

## 3.- Parámetros de entrada

Los procedimientos almacenados pueden recibir y devolver información; para ello se emplean parámetros, de entrada y salida, respectivamente.

Veamos los primeros. Los parámetros de entrada posibilitan pasar información a un procedimiento.

Para que un procedimiento almacenado admita parámetros de entrada se deben declarar variables como parámetros al crearlo. La sintaxis es:

```
 create proc NOMBREPROCEDIMIENTO
```

@NOMBREPARAMETRO TIPO =VALORPORDEFECTO
as SENTENCIAS;

Los parámetros se definen después del nombre del procedimiento, comenzando el nombre con un signo arroba (@). Los parámetros son locales al procedimiento, es decir, existen solamente dentro del mismo. Pueden declararse varios parámetros por procedimiento. La forma es separarlos por comas.

Cuando el procedimiento es ejecutado, deben especificarse valores para cada uno de los parámetros (en el orden que fueron definidos), a menos que se haya definido un valor por defecto, en tal caso, pueden omitirse.

El valor por defecto puede ser "null" o una constante, también puede incluir comodines si el procedimiento emplea "like".

Creamos un procedimiento que recibe el nombre de un autor como parámetro para mostrar todos los libros del autor solicitado:

```
create procedure pa_libros_autor
 @autor varchar(30)
as
 select titulo, editorial,precio
  from libros
  where autor= @autor;
```

El procedimiento se ejecuta colocando "execute" (o "exec") seguido del nombre del procedimiento y un valor para el parámetro:

```
exec pa_libros_autor 'Borges';
```

Ejemplo: Creamos un procedimiento que recibe 2 parámetros, el nombre de un autor y el de una editorial:

```
create procedure pa_libros_autor_editorial
 @autor varchar(30),
 @editorial varchar(20)
as
 select titulo, precio
  from libros
  where autor= @autor and
  editorial=@editorial;
```

El procedimiento se ejecuta colocando "execute" (o "exec") seguido del nombre del procedimiento y los valores para los parámetros separados por comas:

```
exec pa_libros_autor_editorial 'Richard Bach','Planeta';
```

Los valores de un parámetro pueden pasarse al procedimiento mediante el nombre del parámetro o por su posición. La sintaxis anterior ejecuta el procedimiento pasando valores a los parámetros por posición. También podemos emplear la otra sintaxis en la cual pasamos valores a los parámetros por su nombre:

```
exec pa_libros_autor_editorial @editorial='Planeta', @autor='Richard Bach';
```

Cuando pasamos valores con el nombre del parámetro, el orden en que se colocan puede alterarse.

No podríamos ejecutar el procedimiento anterior sin valores para los parámetros. Si queremos ejecutar un procedimiento que permita omitir los valores para los parámetros debemos, al crear el procedimiento, definir valores por defecto para cada parámetro:

```
create procedure pa_libros_autor_editorial2
 @autor varchar(30)='Richard Bach',
 @editorial varchar(20)='Planeta'
as
 select titulo, autor,editorial,precio
  from libros
  where autor= @autor and
  editorial=@editorial;
```

Podemos ejecutar el procedimiento anterior sin enviarle valores, y usará los predeterminados.

Si enviamos un solo parámetro a un procedimiento que tiene definido más de un parámetro sin especificar a qué parámetro corresponde (valor por posición), asume que es el primero. Es decir, SQL Server asume que los valores se dan en el orden que fueron definidos, no se puede interrumpir la secuencia.

Si queremos especificar solamente el segundo parámetro, debemos emplear la sintaxis de paso de valores a parámetros por nombre:

```
exec pa_libros_autor_editorial2 @editorial='Paidos';
```

Podemos emplear patrones de búsqueda en la consulta que define el procedimiento almacenado y utilizar comodines como valores por defecto:

```
create proc pa_libros_autor_editorial3
 @autor varchar(30) = '%',
 @editorial varchar(30) = '%'
as
 select titulo,autor,editorial,precio
```

```
   from libros
   where autor like @autor and
   editorial like @editorial;
```

La sentencia siguiente ejecuta el procedimiento almacenado "pa_libros_autor_editorial3" enviando un valor por posición, se asume que es el primero.

```
exec pa_libros_autor_editorial3 'P%';
```

La sentencia siguiente ejecuta el procedimiento almacenado "pa_libros_autor_editorial3" enviando un valor para el segundo parámetro, para el primer parámetro toma el valor por defecto:

```
exec pa_libros_autor_editorial3 @editorial='P%';
```

También podríamos haber tecleado:

```
exec pa_libros_autor_editorial3 default, 'P%';
```

**4.- Parámetros de salida**

los procedimientos almacenados pueden devolver información; para ello se emplean parámetros de salida. El valor se retorna a quien realizó la llamada con parámetros de salida. Para que un procedimiento almacenado devuelva un valor se debe declarar una variable con la palabra clave "output" al crear el procedimiento:

```
create procedure NOMBREPROCEDIMIENTO
 @PARAMETROENTRADA TIPO =VALORPORDEFECTO,
 @PARAMETROSALIDA TIPO=VALORPORDEFECTO output
 as
  SENTENCIAS
  select @PARAMETROSALIDA=SENTENCIAS;
```

Los parámetros de salida pueden ser de cualquier tipo de datos, excepto text, ntext e image.

Creamos un procedimiento almacenado al cual le enviamos 2 números y retorna el promedio:

```
create procedure pa_promedio
  @n1 decimal(4,2),
  @n2 decimal(4,2),
  @resultado decimal(4,2) output
```

```
  as
   select @resultado=(@n1+@n2)/2;
```

Al ejecutarlo también debe emplearse "output":

```
 declare @variable decimal(4,2)
  execute pa_promedio 5,6, @variable output
  select @variable;
```

Declaramos una variable para guardar el valor devuelto por el procedimiento; ejecutamos el procedimiento enviándole 2 valores y mostramos el resultado.

La instrucción que realiza la llamada al procedimiento debe contener un nombre de variable para almacenar el valor retornado.

Creamos un procedimiento almacenado que muestre los títulos, editorial y precio de los libros de un determinado autor (enviado como parámetro de entrada) y nos retorne la suma y el promedio de los precios de todos los libros del autor enviado:

```
create procedure pa_autor_sumaypromedio
  @autor varchar(30)='%',
  @suma decimal(6,2) output,
  @promedio decimal(6,2) output
  as
   select titulo,editorial,precio
   from libros
   where autor like @autor
  select @suma=sum(precio)
   from libros
   where autor like @autor
  select @promedio=avg(precio)
   from libros
   where autor like @autor;
```

Ejecutamos el procedimiento y vemos el contenido de las variables en las que almacenamos los parámetros de salida del procedimiento:

```
declare @s decimal(6,2), @p decimal(6,2)
 execute pa_autor_sumaypromedio 'Richard Bach', @s output, @p output
 select @s as total, @p as promedio;
```