

Procedimientos almacenados

Contenido

Introducción.....	2
Trabajar con procedimientos almacenados	4
1. Crear procedimientos almacenados.....	4
2. Modificar un procedimiento almacenado.....	4
3. Borrar procedimientos almacenados.....	4
4. Procedimientos almacenados con parámetros de entrada.....	4
5. Ejecutar procedimientos almacenados.....	5
6. Procedimientos almacenados con parámetros de salida	5
7. Procedimientos almacenados que devuelven un valor.....	6
8. Recompilación de procedimientos almacenados	7
9. Identificar la existencia de procedimientos almacenados	7
10. Renombrar procedimientos almacenados	8

Introducción

Se define un Procedimiento Almacenado (o *Stored Procedure*) como una subrutina, procedimiento o programa, que se encuentra disponible para las aplicaciones que acceden a una BBDD relacional. Se denominan “almacenados” porque en lugar de estar integrados con el resto del código de la aplicación se encuentran **almacenados** en la propia BBDD. Son muy utilizados a la hora de crear aplicaciones que usen bases de datos, aunque también se utilizan para tareas de administración.

Sus funciones principales son:

- **Organizar** las consultas de una manera clara y lógica.
- **Separar** parte de la lógica de la aplicación (también llamada lógica de negocio) del resto del código.
- Trasladar y **centralizar** dicha lógica al servidor de BBDD. De esta forma el único tráfico entre el servidor de aplicaciones y el servidor de BBDD es una única llamada para su ejecución, en lugar de una llamada por cada transacción involucrada, y, si fuera necesario, el resultado del procedimiento.
- Al ejecutarse en el servidor de BBDD, que generalmente es distinto al servidor web, se distribuye la carga, mejorando el **rendimiento** final de la aplicación.
- Implementar de forma **integrada** funcionalidades de la aplicación. Por ejemplo: obtener todos los productos con determinadas características, generar las opciones de un menú, añadir un producto al carrito o modificar su cantidad, etc.
- **Encapsular** una serie de instrucciones, proporcionando un nivel de abstracción.

Aunque pueda parecer que tienen similitudes con las vistas, son mucho más complejos y versátiles que estas. Una de las principales diferencias es que a los procedimientos almacenados se les pueden incluir **parámetros de entrada**.

Pueden utilizar **todos los comandos** vistos hasta ahora, tanto **LDD** (create, alter, drop), como **LMD** (select, insert, update, delete). Incluso se pueden incluir sentencias **LCD** (grant, revoke). E incluso pueden incluir llamadas a otros Procedimientos Almacenados (exec).

A veces se utilizan únicamente de forma sencilla para hacer pequeñas SELECT, INSERT o UPDATE. Otras veces se generan Procedimientos Almacenados muy complicados, con gran número de instrucciones, con idea de que se puedan usar y reutilizar mediante una simple llamada al mismo. La idea es hacer un paquete que **encapsule** muchas sentencias SQL. Por ejemplo: generar un informe de múltiples tipos de productos de una tienda, con su número de ventas, estadísticas detalladas y agrupadas y cambios de moneda.

Utilizar Procedimientos Almacenados es la mejor manera de otorgar funcionalidades a una aplicación. Se debe pensar en ellos como simples instrucciones que una aplicación puede utilizar. Por ejemplo: añadir un nuevo usuario a la aplicación, añadir un nuevo producto a la

tienda o al carrito, calcular una nómina, etc. De esta forma no es necesario **reescribir** las mismas consultas una y otra vez en diferentes partes de la aplicación. Esto hace que se reduzca la probabilidad de que los datos se corrompan por un uso erróneo, lo que asegura la integridad y consistencia de los datos.

Proporcionan **un punto de acceso común a los datos**, algo muy útil en equipos de desarrollo amplios (más de 1 desarrollador), lo que también facilita el mantenimiento. En lugar de que cada parte del programa acceda por su cuenta a los datos, alguien del equipo se encarga de desarrollar ese acceso a los datos mediante Procedimientos Almacenados y al resto de programadores se les facilita permiso de ejecución sobre esos Procedimientos, despreocupándose de cómo funcionan y pudiendo centrarse en sus propios desarrollos.

Otro aspecto muy importante acerca del uso de procedimientos almacenados en el diseño de aplicaciones es que pueden *prevenir* el uso de **ataques por inyección de SQL**, al estar predefinidos los campos de la aplicación para aceptar únicamente valores válidos para dicha consulta. Eso no es así siempre y para cualquier procedimiento almacenado.

Por lo tanto, otro punto importante es que también se pueden, y deben, otorgar **permisos** sobre los Procedimientos Almacenados, de la misma manera a como se hace para cualquier otro tipo de objeto. De esta manera los datos accedidos por el procedimiento solo estarán disponibles para los usuarios requeridos. En este caso el permiso que más interesa será el de EXECUTE.

Ya hay creados muchos Procedimientos Almacenados por defecto, principalmente para tareas de administración y/o depuración, que vienen identificados con el prefijo “sp_”. Algunos de los más interesantes son:

- sp_who2
- sp_help
- sp_helptext
- sp_helpindex
- sp_helpconstraint
- sp_depends
- sp_spaceused
- sp_databases
- sp_tables
- sp_rename
- sp_executesql

En la práctica se recomienda utilizar una nomenclatura consistente para todos los procedimientos almacenados (y para todos los objetos de la BBDD), lo que implica utilizar el mismo prefijo para todos. Se recomienda NO utilizar el prefijo “sp_”, utilizado por el sistema.

Los Procedimientos Almacenados pueden no tener ninguna salida, devolver una serie de datos o devolver un mensaje, por ejemplo, para informar del estado de la ejecución.

Es habitual en el uso de procedimientos almacenados complejos crear tablas temporales (#) en los que insertar los valores resultantes de inner joins entre tablas, incluso de diferentes bases de datos. Estas tablas temporales son borradas al final del procedimiento, bien

explícita (DROP) o automáticamente al finalizar su ejecución. Siempre es recomendable realizar un borrado explícito cuando sabemos que no se van a necesitar más los datos.

Trabajar con procedimientos almacenados

1. Crear procedimientos almacenados

```
CREATE PROCEDURE <Nombre>
AS
<Comandos>
```

```
CREATE PROCEDURE PA_MOSTRAR_PERSONAS
AS
SELECT nombre, apellidos
FROM Persona
```

2. Modificar un procedimiento almacenado

```
ALTER PROCEDURE <Nombre>
AS
<Comandos>
```

```
ALTER PROCEDURE PA_MOSTRAR_PERSONAS
AS
SELECT nombre, apellidos, teléfono
FROM Personas
```

Aunque a veces en lugar de modificarlo directamente se borra el existente y se vuelve a crear. (if exists drop procedure. Go. Create Procedure)

3. Borrar procedimientos almacenados

```
DROP PROCEDURE <Nombre>
```

```
DROP PROC <Nombre>
```

4. Procedimientos almacenados con parámetros de entrada

```
CREATE PROCEDURE <Nombre>
    @parametro TIPO          < =Valor Opcional. Valor por defecto>
AS
    <Comandos>
```

```
CREATE PROCEDURE PA_BuscaDatos
    @nombre VARCHAR(50),
    @edad int      [=18]
AS
    SELECT dni, tlf FROM Personas
    WHERE nombre = @nombre AND
           edad = @edad
```

Se pueden especificar, de forma opcional, valores por defecto a los parámetros de entrada, de manera que la ejecución tome dicho valor si no se le pasa ninguno. Puede mezclarse el uso de parámetros de entrada con valores iniciales y sin ellos. Sin embargo, para poder obviar el valor de los parámetros definidos con valor inicial deberán estar definidos al final de la lista de declaraciones de parámetros.

NO se pueden utilizar los parámetros de entrada para especificar los nombres de tablas, columnas o cláusulas ORDER BY.

5. Ejecutar procedimientos almacenados

Si no tiene parámetros de entrada:

```
EXECUTE <Nombre_pa>
EXEC <Nombre_pa>
<Nombre_pa>
```

Si tiene parámetros de entrada:

```
EXECUTE <Nombre_pa> @nombre_parametro = 'valor'
EXEC <Nombre_pa> 'valor'
<Nombre_pa> 'valor'
```

Si se especifican las variables de los parámetros de entrada pueden indicarse en cualquier orden, sin embargo, si se pasan únicamente los valores (sin el nombre del parámetro) deberán indicarse en el nombre exacto en el que se han definido dentro del procedimiento.

6. Procedimientos almacenados con parámetros de salida

Devuelve un resultado de una consulta como resultado de salida. Es decir, devuelve un valor que está **almacenado** en la base de datos mediante un parámetro al que le indicamos que es de salida. Posteriormente deberemos guardar en dicho parámetro el valor que queremos devolver, es decir, el resultado de una consulta.

```
CREATE PROCEDURE PA_MostrarInscripcion
    @resultado VARCHAR(50) OUTPUT
AS
    SET resultado = (SELECT inscripcion
                     FROM Enfermo
                     WHERE Apellido = 'Serrano V.')
```

Y obtenemos su resultado al ejecutar:

```
DECLARE @outputRes VARCHAR(50)
EXEC PA_MostrarInscripcion @outputRes OUTPUT
SELECT @outputRes
```

Este método es útil si únicamente extraemos un valor. Si queremos obtener varios valores, utilizaremos varios parámetros de salida. Por lo tanto, una manera más completa para obtener los resultados utilizando una única consulta sería la siguiente:

```
CREATE PROCEDURE PA_MostrarInscripcion
    @inscrip VARCHAR(50) OUTPUT,
    @direccion VARCHAR(25) OUTPUT
AS
    SELECT @inscrip = inscripcion, @direccion = direccion
    FROM Enfermo
    WHERE Apellido = 'Serrano V.')
```

Y obtenemos sus resultados al ejecutar:

```
DECLARE @outputRes1 VARCHAR(50)
DECLARE @outputRes2 VARCHAR(25)
EXEC PA_MostrarInscripcion @outputRes1 OUTPUT, @outputRes2 OUTPUT
SELECT @outputRes1, @outputRes2
```

7. Procedimientos almacenados que devuelven un valor

En este caso devolvemos un valor, el que queramos, que no tiene por qué ser un dato de la base de datos, suele utilizarse para devolver un valor para notificar el estado de ejecución. Para devolver valores de la BBDD se recomienda el uso de OUTPUT.

```
CREATE PROCEDURE PA_ContarEnfermos
AS
    DECLARE @resultado AS INT
    SET @resultado = (SELECT COUNT(apellido)
                     FROM enfermo)
    RETURN @resultado
```

Y para obtener el resultado:

```
DECLARE @returnvalue INT
EXEC @returnvalue = PA_ContarEnfermos
SELECT @returnvalue
```

Aunque este método sirve para devolver valores (un único valor) hay que tener en cuenta que, en la práctica, se suele utilizar para informar acerca del estado de la ejecución del procedimiento almacenado. Los únicos valores que se pueden devolver son de tipo entero (*int*), es decir no se pueden devolver cadenas de caracteres y demás tipos de datos.

Adicionalmente se pueden declarar variables dentro de un Procedimiento Almacenado. Estas difieren de los parámetros de entrada/salida en que no serán solicitados ni devueltos al ejecutar el procedimiento. Se eliminarán al terminar la ejecución.

```
CREATE PROCEDURE MiProcedimiento
    @Param_entrada TIPO
AS
    DECLARE @variable TIPO
    SELECT ...
```

8. Recompilación de procedimientos almacenados

Los procedimientos almacenados se compilan con la primera ejecución, resultando en la creación de un plan de ejecución optimizado que el sistema guarda en la caché y que será utilizado en las siguientes ejecuciones, mejorando su rendimiento.

Cuando un procedimiento se compila por primera vez o se vuelve a compilar, el plan de consulta del procedimiento se optimiza para el estado actual de la base de datos y sus objetos. Si una base de datos experimenta cambios significativos en sus datos o su estructura, al volver a compilar un procedimiento se actualiza y optimiza el plan de consulta del procedimiento para tener en cuenta esos cambios. Esto puede mejorar el rendimiento de procesamiento del procedimiento

Se puede utilizar esta opción de varias maneras:

- Añadiendo WITH RECOMPILE en la definición del procedimiento o al realizar su ejecución

```
ALTER PROCEDURE PA_nombre (@string VARCHAR(50)) WITH RECOMPILE AS <...>
```

- Mediante el procedimiento almacenado del sistema sp_recompile

```
EXEC SP_RECOMPILE N'dbo.Pa_Nombre'
```

9. Identificar la existencia de procedimientos almacenados

En los Procedimientos Almacenados se pueden incluir prácticamente todas las sentencias vistas durante el curso, incluyendo todas las que se han visto para la creación de scripts: declaración de variables, sentencias de control de flujo (if else, while, case), comprobaciones de las tablas del sistema...

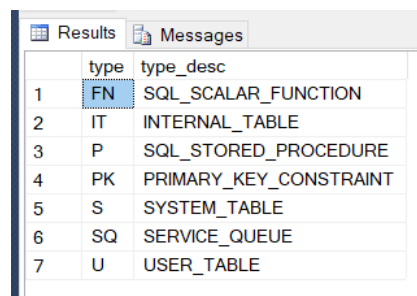
Por lo tanto, como los PA son objetos almacenados en la BBDD (al igual que lo son, por ejemplo, las tablas o las vistas), se pueden utilizar las funciones aprendidas hasta ahora. Por ejemplo, para comprobar si un PA existe podemos hacer uso de las funciones OBJECT_ID y OBJECT_NAME.

```
IF OBJECT_ID('PA_Nombre', 'P') IS NOT NULL
    DROP PROCEDURE 'PA_Nombre'
```

El Segundo parámetro pasado a la función `OBJECT_ID` es opcional y consiste en el tipo de objeto que se busca dentro de la base de datos. Si se especifica nos estaremos asegurando de encontrar el objeto esperado, ya que, en principio, nada impide que pueda haber otros objetos con el mismo nombre. Esto nos demuestra la importancia de nombrar a cada objeto correctamente, por ejemplo, utilizando un mismo prefijo para todos los procedimientos almacenados, otro diferente para todas las funciones, otro para las tablas, etc...

Para comprobar los tipos de objetos que podemos tener y su correspondiente código (que sería el que pasaríamos como segundo parámetro a la función `OBJECT_ID`), podemos consultar la vista de sistema `sys.objects` de la siguiente manera:

```
SELECT DISTINCT [type], [type_desc] FROM sys.objects
```



	type	type_desc
1	FN	SQL_SCALAR_FUNCTION
2	IT	INTERNAL_TABLE
3	P	SQL_STORED_PROCEDURE
4	PK	PRIMARY_KEY_CONSTRAINT
5	S	SYSTEM_TABLE
6	SQ	SERVICE_QUEUE
7	U	USER_TABLE

10. Renombrar procedimientos almacenados

Mediante la sentencia de modificación de procedimientos almacenados (`ALTER`) no se pueden renombrar, puesto que al indicarle un nombre diferente al original el sistema lo interpreta como un procedimiento almacenado diferente. Por lo tanto, para cambiar el nombre a un procedimiento almacenado nos encontramos con dos opciones:

- Eliminar el procedimiento antiguo y crearlo de nuevo:

```
IF EXISTS OBJECT_ID('PA_AntiguoNombre', 'P')
    DROP PROCEDURE 'PA_AntiguoNombre'

CREATE PROCEDURE PA_NuevoNombre
AS <...>
```

- Utilizar el procedimiento almacenado de sistema `'sp_rename'`:

Cabe destacar que es importante indicar el esquema en el nombre del procedimiento almacenado original.

```
EXEC sp_rename 'dbo.PA_AntiguoNombre', 'PA_NuevoNombre'
```