

Procedimientos almacenados

Bases de datos

¿Qué es un procedimiento almacenado?

Se define un PA 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, por ejemplo, en un servidor web, se encuentran **almacenados** en la propia BBDD, lo que tiene varias ventajas.

Se utilizan muy frecuentemente para crear aplicaciones, pero también tienen un gran uso en tareas de administración.

¿Cuál es su función?

- **Organizar** las consultas de una manera clara y lógica.
- **Separar** parte de la lógica de la aplicación (o lógica de negocio) del resto del código.
- **Trasladar y centralizar** dicha lógica al servidor de BBDD, reduciendo el tráfico entre el servidor de aplicaciones y el de BBDD a **una única llamada**, en lugar de requerir una llamada para cada transacción involucrada. Y, si fuera necesario, también 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**.
- Implementar **funcionalidades** de la aplicación de forma integrada. P. ej. obtener un listado de productos, las opciones de un menú, añadir un producto al carro de la compra...
- **Encapsular** una serie de instrucciones, proporcionando un nivel de abstracción.

¿Cómo se pueden construir?

Pueden utilizar prácticamente todos los comandos vistos hasta ahora:

- **LDD**: create, alter, drop...
- **LMD**: select, insert, update, delete...
- **LCD**: grant, revoke...

Incluso pueden incluir llamadas a otros procedimientos almacenados (EXEC)

Pueden tener parámetros de entrada y de salida, lo que les confiere la diferencia más importante con respecto a las vistas.

¿Cómo se suelen utilizar?

- Se busca encapsular muchas sentencias que puedan ser reutilizadas mediante una simple llamada. Por ejemplo, crear un informe de múltiples productos de una tienda (ventas, estadísticas agrupadas, cambios de moneda...)
- Es la mejor manera de otorgar funcionalidades a una aplicación. Se puede pensar en ellos como “simples” instrucciones que una aplicación puede utilizar. Por ejemplo, añadir un nuevo producto al catálogo o al carrito de la compra, calcular una nómina, etc...
- Reduce la posibilidad de corrupción en los datos y pueden ayudar a prevenir ataques por inyección de SQL.

¿Cómo se suelen utilizar?

- Proporcionan un punto de acceso común a los datos, importante en equipos de desarrollo amplios, y que también facilita el mantenimiento.
 - Tan sólo una parte del equipo se encarga de desarrollar el acceso a los datos, a través de procedimientos almacenados.
 - El resto se les facilita el permiso de ejecución sobre los procedimientos, abstrayéndose de cómo funcionan y centrándose en sus propios desarrollos.
- Por lo tanto, se pueden, y se deben, otorgar permisos (de ejecución, modificación, etc.) sobre los procedimientos almacenados al igual que sobre el resto de objetos de una base de datos.

¿Cómo se suelen utilizar?

- Existen multitud de procedimientos almacenados ya incluidos por defecto en una base de datos. Principalmente se utilizan para tareas de administración y van precedidos por el prefijo “sp_”. (sp_who2, sp_help, sp_helptext, sp_helpindex, sp_helpconstraint, sp_depends, sp_spaceused, sp_databases, sp_tables, sp_rename, sp_executesql).
- 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 uniones de tablas, incluso de diferentes bases de datos. Estas tablas son borradas al final del procedimiento, bien explícita (DROP) o automáticamente al finalizar su ejecución.

Crear procedimientos almacenados

- Crear:

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

```
CREATE PROCEDURE PA_Mostrar_Empleados  
AS  
    SELECT Nombre, Apellidos  
    FROM Empleados
```

- Modificar:

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

```
ALTER PROCEDURE PA_Mostrar_Empleados  
AS  
    SELECT Tratamiento, Nombre, Apellidos, Cargo  
    FROM Empleados
```

- Borrar:

```
DROP PROCEDURE <Nombre>
```

```
DROP PROCEDURE PA_Mostrar_Empleados
```


PAs con parámetros de entrada

```
CREATE PROCEDURE <Nombre>  
    @parametro TIPO  
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 valores por defecto a los parámetros de entrada, de manera que la ejecución tome dicho valor si no se le pasa ninguno.

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

Ejecución de PAs

- Si no tiene parámetros de entrada

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

```
EXECUTE PA_Mostrar_Empleados  
EXEC PA_Mostrar_Empleados  
PA_Mostrar_Empleados
```

- Si tiene parámetros de entrada

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

```
EXECUTE PA_BuscaDatos 'Pedro', '23'  
EXEC PA_BuscaDatos 'Pedro', '23'  
PA_BuscaDatos 'Pedro', '23'
```

PAs con parámetros de salida

Devuelve un resultado de una consulta SQL como resultado de salida. Es decir, devuelve un valor que está **almacenado** en la base de datos a través de un parámetro al que le indicamos que es de salida.

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

Y obtenemos su resultado al ejecutar:

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

PAs 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 obtenemos su resultado al ejecutar:

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

Recompilar procedimientos almacenados

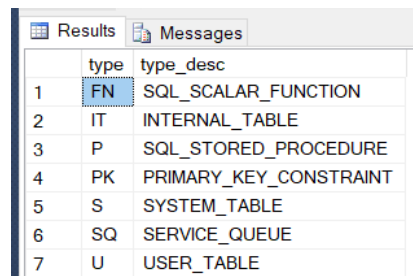
- Los PA se compilan durante su primera ejecución, durante la que se crea un plan de ejecución optimizado que el sistema guarda para posteriores ejecuciones. Esto mejora su rendimiento.
- Cuando un PA se compila, su plan de ejecución se optimiza para el estado actual de la base de datos y sus objetos existentes. Si una base de datos experimenta cambios significativos en sus datos o en su estructura, puede ser buena idea recompilar los procedimientos almacenados afectados por esos cambios.
- Se puede hacer de dos maneras:
 - Añadiendo WITH RECOMPILE en la definición
`ALTER PROCEDURE PA_Nombre (@string varchar(50)) WITH RECOMPILE AS <...>`
 - Mediante el procedimiento almacenado de sistema *sp_recompile*:
`EXEC SP_RECOMPILE N'dbo.PA_Nombre'`

Procedimientos almacenados

- En los Procedimientos Almacenados se pueden incluir todas las sentencias vistas hasta ahora, incluyendo las vistas con scripts, declaración de variables, sentencias de control de flujo (if else, while, case), comprobaciones de las tablas del sistema...
- Como los PA son objetos, se pueden utilizar las funciones vistas hasta ahora. Por ejemplo, para comprobar si un PA existe podemos hacer uso de las funciones OBJECT_ID y OBJECT_NAME.

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

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



The screenshot shows a SQL Server Results window with two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with two columns: 'type' and 'type_desc'. The table contains seven rows of data, with the first row highlighted in blue. The 'type' column values are FN, IT, P, PK, S, SQ, and U. The 'type_desc' column values are SQL_SCALAR_FUNCTION, INTERNAL_TABLE, SQL_STORED_PROCEDURE, PRIMARY_KEY_CONSTRAINT, SYSTEM_TABLE, SERVICE_QUEUE, and USER_TABLE.

	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

Renombrar Procedimientos Almacenados

- Para renombrar un procedimiento almacenado tenemos dos opciones:
 - Eliminar el antiguo y crear el 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'

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