

SQL SERVER: OPTIMIZANDO CON PROCEDIMIENTOS ALMACENADOS

Carlos Poveda

Roberto Lasso

Cristian Diaz

INTRODUCCIÓN

- La demo que se desarrolló es una pequeña tienda llamada “SQL Store”, utilizando HTML y JavaScript para el frontend, el backend se utilizo Node.js y la base de datos en SQL Server.

- El objetivo principal es demonstrar como se optimizo el acceso a datos usando procedimientos almacenados en SQL Server, en lugar de ejecutarlos directamente desde backend.



SQL SERVER COMO MOTOR DE EJECUCIÓN



Los procedimientos almacenados permiten que las operaciones de acceso a datos se ejecuten directamente dentro de SQL Server. En este proyecto, acciones como la obtención de productos, la consulta de historial y el procesamiento de compras se realizan exclusivamente mediante procedimientos almacenados, delegando la ejecución de las operaciones al servidor en lugar de implementarlas como consultas directas en la aplicación.

¿QUÉ ES UN PROCEDIMIENTO ALMACENADO?

Los procedimientos almacenados permiten encapsular la lógica de acceso a datos dentro de SQL Server, evitando consultas directas desde la aplicación como SELECT o INSERT expuestos. En este proyecto, todas las operaciones críticas se ejecutan mediante procedimientos, lo que centraliza la lógica en el servidor. Además, el uso de parámetros tipados reduce el riesgo de inyección SQL y mejora la organización del sistema al separar claramente la capa de aplicación de la lógica de base de datos.



PROCEDIMIENTOS ALMACENADOS

- Los procedimientos almacenados permiten encapsular la lógica de acceso a datos dentro de SQL Server, evitando consultas directas desde la aplicación como SELECT o INSERT expuestos. Esto mejora el rendimiento al reutilizar planes de ejecución, incrementa la seguridad al restringir acceso directo a tablas y facilita el mantenimiento al centralizar la lógica en un solo punto. Además, reducen el riesgo de SQL Injection al trabajar con parámetros tipados en lugar de concatenaciones dinámicas.

VENTAJAS CLAVE EN LA OPTIMIZACIÓN

- Reducción de Llamadas a la Base de Datos: Operaciones como el procesamiento de compra se ejecutan en una sola llamada al procedimiento, evitando múltiples consultas desde la aplicación.
- Centralización de la Lógica: Las reglas de negocio se ejecutan dentro de SQL Server y no en el backend, reduciendo complejidad en la aplicación.
- Uso de Parámetros Tipados: Se evita la concatenación dinámica de consultas, mejorando organización y reduciendo riesgos de inyección SQL.
- Reutilización del Plan de Ejecución: Al utilizar procedimientos almacenados, SQL Server puede reutilizar planes de ejecución generados previamente.



ELIMINACIÓN DE CONSULTAS DIRECTAS

El backend no ejecuta sentencias SELECT, INSERT o UPDATE manuales. En su lugar, utiliza instrucciones EXEC para invocar procedimientos como ObtenerProductos, ObtenerVentasCliente y ProcesarCompra. Esto centraliza la lógica en SQL Server y reduce el acoplamiento entre aplicación y base de datos.

OPTIMIZACIÓN MEDIANTE PARÁMETROS

Las consultas que reciben datos externos utilizan parámetros tipados en lugar de concatenaciones dinámicas. Esto mejora seguridad y permite a SQL Server manejar la ejecución de manera más estructurada.

Ejemplo aplicado:

El procedimiento ObtenerVentasCliente recibe @Email como parámetro en lugar de construir una consulta manual.

OPTIMIZACIÓN CON TABLE-VALUED PARAMETER

El procesamiento de compra utiliza un parámetro estructurado para enviar múltiples productos en una sola ejecución. En lugar de enviar múltiples INSERT desde la aplicación, se envía una tabla completa como parámetro al procedimiento `ProcesarCompra`.

Esto reduce:

- Número de llamadas a la base de datos
- Código repetitivo en el backend
- Complejidad de la capa de aplicación

CENTRALIZACIÓN DE OPERACIONES COMPLEJAS

El procedimiento `ProcesarCompra` concentra múltiples acciones dentro de SQL Server, como inserciones y actualizaciones relacionadas con una venta. Esto evita que el backend tenga que ejecutar múltiples consultas separadas. La optimización en este caso se logra al ejecutar la lógica directamente en el motor de base de datos en lugar de distribuirla en varias llamadas externas.

CONCLUSION

- La optimización implementada en este proyecto consiste en trasladar la lógica crítica a procedimientos almacenados, utilizar parámetros tipados y reducir múltiples operaciones a ejecuciones únicas dentro de SQL Server. No se trata de una optimización avanzada de bajo nivel, sino de una optimización estructural basada en buenas prácticas de arquitectura.





GRACIAS