
Documentación Técnica: Monitoreo y Rendimiento en SQL Server (Semana 13)

Objetivo General: Optimizar la eficiencia del motor de base de datos SQL Server, asegurar la estabilidad del sistema y prevenir cuellos de botella mediante herramientas de diagnóstico y buenas prácticas de desarrollo T-SQL.

1. Análisis de rendimiento con SQL Profiler y Extended Events

Definición

Herramientas para capturar y monitorear la actividad del servidor en tiempo real.

- **SQL Profiler:** Herramienta gráfica tradicional (actualmente en desuso/deprecated) para trazar eventos.
- **Extended Events (XEEvents):** Sistema de monitoreo ligero, escalable y de bajo impacto en el rendimiento, diseñado para reemplazar a Profiler. Permite capturar datos detallados para la solución de problemas.

Script de Ejemplo: Extended Events

Este script crea una sesión de eventos extendidos para capturar consultas que tardan más de 1 segundo (1000000 microsegundos).

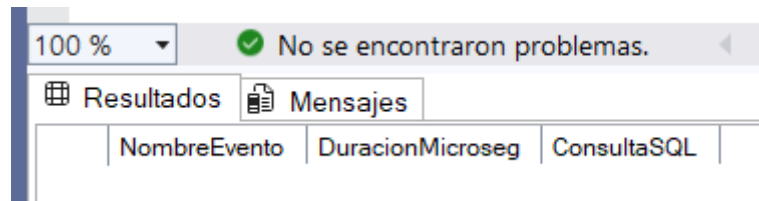
SQL

```
-- 1. Crear la sesión de eventos
CREATE EVENT SESSION [CapturaConsultasLentas] ON SERVER
ADD EVENT sqlserver.sql_statement_completed(
    ACTION(sqlserver.database_name, sqlserver.client_app_name,
    sqlserver.username)
    WHERE ([duration] > (1000000)) -- Duración mayor a 1 segundo
)
ADD TARGET package0.event_file(SET filename=N'ConsultasLentas.xel');
GO

-- 2. Iniciar la sesión
ALTER EVENT SESSION [CapturaConsultasLentas] ON SERVER STATE = START;
GO

-- 3. Para consultar los datos capturados (XML parseado)
SELECT
    event_data.value('(event/@name)[1]', 'varchar(50)') AS NombreEvento,
    event_data.value('(event/data[@name="duration"]/value)[1]', 'bigint') AS
DuracionMicroseg,
    event_data.value('(event/data[@name="statement"]/value)[1]',
'nvarchar(max)') AS ConsultaSQL
FROM (
    SELECT CAST(event_data AS XML) AS event_data
```

```
FROM sys.fn_xe_file_target_read_file('ConsultasLentas*.xel', null, null,
null)
) AS XEventData;
```



2. Estadísticas e índices (Creación, fragmentación, mantenimiento)

Definición

- **Índices:** Estructuras en disco (B-Tree) que aceleran la recuperación de filas. Pueden ser *Clustered* (ordenan físicamente los datos) o *Non-Clustered*.
- **Estadísticas:** Objetos binarios que describen la distribución de los datos en una columna. El optimizador de consultas las usa para estimar cuántas filas devolverá una consulta (cardinalidad).
- **Fragmentación:** Ocurre cuando el orden lógico de las páginas del índice no coincide con el orden físico, degradando el rendimiento de E/S.

Script de Ejemplo: Mantenimiento de Índices

Identifica índices fragmentados y reconstruye aquellos con más del 30% de fragmentación.

SQL

```
-- Verificar fragmentación
SELECT
    dbschemas.[name] as 'Schema',
    dbtables.[name] as 'Table',
    dbindexes.[name] as 'Index',
    indexstats.avg_fragmentation_in_percent
FROM sys.dm_db_index_physical_stats (DB_ID(), NULL, NULL, NULL, NULL) AS
indexstats
INNER JOIN sys.indexes AS dbindexes ON indexstats.object_id =
dbindexes.object_id
INNER JOIN sys.objects AS dbtables ON indexstats.object_id =
dbtables.object_id
INNER JOIN sys.schemas AS dbschemas ON dbtables.schema_id =
dbschemas.schema_id
WHERE indexstats.database_id = DB_ID() AND
indexstats.avg_fragmentation_in_percent > 30;

-- Ejemplo de Reconstrucción (Rebuild) y Actualización de Estadísticas
USE QhātuPERU;
GO
ALTER INDEX [IX_NombreIndice] ON [Esquema].[Tabla] REBUILD;
GO
UPDATE STATISTICS [Esquema].[Tabla] WITH FULLSCAN;
```

3. Administración de transacciones y bloqueos

Definición

- **Transacción:** Unidad lógica de trabajo que debe cumplir las propiedades ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad).
- **Bloqueos (Locks):** Mecanismo para gestionar el acceso concurrente a los recursos. Un bloqueo excesivo puede causar *Blocking* (espera) o *Deadlocks* (abrazo mortal).

Script de Ejemplo: Monitoreo de Bloqueos

Script para identificar qué sesión está bloqueando a otras en este momento.

SQL

```
-- Ver procesos bloqueados actualmente
SELECT
    r.session_id as [Sesion_Bloqueada],
    r.blocking_session_id as [Sesion_Bloqueadora],
    r.wait_type as [Tipo_Espera],
    r.wait_time as [Tiempo_Espera_ms],
    t.text as [Consulta_Ejecutandose]
FROM sys.dm_exec_requests r
CROSS APPLY sys.dm_exec_sql_text(r.sql_handle) t
WHERE r.blocking_session_id <> 0;

-- Ejemplo de Transacción Segura con Manejo de Errores
BEGIN TRY
    BEGIN TRANSACTION;
        -- Operaciones DML
        UPDATE Inventario SET Cantidad = Cantidad - 1 WHERE ProductoID =
100;
        INSERT INTO Ventas (ProductoID, Fecha) VALUES (100, GETDATE());
    COMMIT TRANSACTION;
END TRY
BEGIN CATCH
    IF @@TRANCOUNT > 0
        ROLLBACK TRANSACTION;
    PRINT 'Error: ' + ERROR_MESSAGE();
END CATCH;
```

4. Análisis de planes de ejecución

Definición

Es la representación visual o XML de los pasos que el motor de SQL Server estima necesarios para ejecutar una consulta. Permite identificar operaciones costosas como *Table Scans* (escaneos completos), *Key Lookups* o *Sorts* innecesarios.

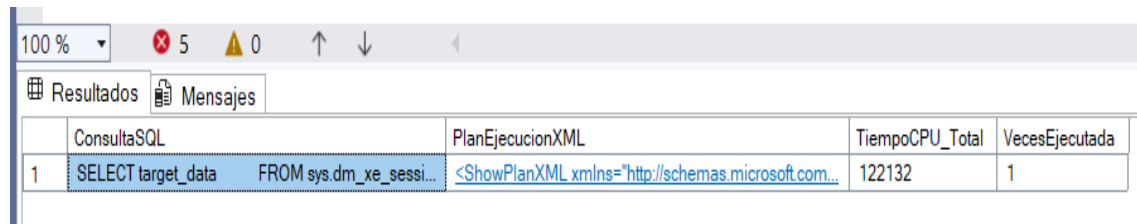
Script de Ejemplo: Obtener Planes en Caché

Recupera los planes de ejecución de las consultas que más CPU han consumido.

SQL

-- Top 5 consultas por consumo de CPU con su Plan de Ejecución XML

```
SELECT TOP 5
    st.text AS ConsultaSQL,
    qp.query_plan AS PlanEjecucionXML,
    qs.total_worker_time AS TiempoCPU_Total,
    qs.execution_count AS VecesEjecutada
FROM sys.dm_exec_query_stats qs
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) st
CROSS APPLY sys.dm_exec_query_plan(qs.plan_handle) qp
ORDER BY qs.total_worker_time DESC;;
```



| | ConsultaSQL | PlanEjecucionXML | TiempoCPU_Total | VecesEjecutada |
|---|--|---|-----------------|----------------|
| 1 | SELECT target_data FROM sys.dm_xe_sessi... | <ShowPlanXML xmlns="http://schemas.microsoft.com..." | 122132 | 1 |

5. Optimización de consultas T-SQL

Definición

Práctica de escribir código SQL eficiente. Se enfoca en reducir E/S y uso de CPU.

- **SARGable (Search ARGument ABLE):** Escribir cláusulas `WHERE` que permitan al motor usar índices (ej. evitar funciones en columnas indexadas).
- **Evitar:** `SELECT *`, cursores innecesarios y subconsultas correlacionadas complejas.

Script de Ejemplo: SARGable vs No-SARGable

Comparación de cómo buscar por fecha correctamente para aprovechar índices.

SQL

-- NO OPTIMIZADO (No SARGable)

-- El motor debe escanear toda la tabla porque aplica la función `YEAR` a cada fila.

```
SELECT OrderID, OrderDate
FROM Sales.Orders
WHERE YEAR(OrderDate) = 2023;
```

-- OPTIMIZADO (SARGable)

-- El motor puede buscar directamente en el rango del índice de fechas.

```
SELECT OrderID, OrderDate
FROM Sales.Orders
WHERE OrderDate >= '2023-01-01' AND OrderDate < '2024-01-01';
```

6. Control de recursos con Resource Governor

Definición

Característica de SQL Server (Enterprise Edition) que permite limitar y gestionar el consumo de recursos (CPU y Memoria) para diferentes cargas de trabajo o grupos de usuarios, evitando que una consulta pesada sature todo el servidor.

Script de Ejemplo: Configuración Básica

Limitar un grupo de trabajo al 20% del uso máximo de CPU.

```
SQL
USE master;
GO
-- 1. Crear un Pool de Recursos (Límite físico)
CREATE RESOURCE POOL [PoolReportes]
WITH (MAX_CPU_PERCENT = 20);
GO

-- 2. Crear un Grupo de Carga de Trabajo asignado al Pool
CREATE WORKLOAD GROUP [GrupoReportes]
USING [PoolReportes];
GO

-- 3. Crear Función Clasificadora (Decide quién va a qué grupo)
CREATE FUNCTION dbo.ClasificadorDeUsuarios()
RETURNS SYSNAME WITH SCHEMABINDING
AS
BEGIN
    DECLARE @NombreGrupo SYSNAME;
    IF SUSER_NAME() = 'UsuarioReportes'
        SET @NombreGrupo = 'GrupoReportes';
    ELSE
        SET @NombreGrupo = 'default';
    RETURN @NombreGrupo;
END;
GO

-- 4. Activar Resource Governor con la función
ALTER RESOURCE GOVERNOR WITH (CLASSIFIER_FUNCTION =
dbo.ClasificadorDeUsuarios);
ALTER RESOURCE GOVERNOR RECONFIGURE;
GO
```
