UNIVERSIDAD PERUANA LOS ANDES FACULTAD DE INGENIERÍA

Asignatura: Base de Datos II

Docente: Dr. Raúl Enrique Fernández Bejarano

Alumno: SORIANO TIMOTEO Joel Kevin

Ciclo: V

huancayo-Perú-2025

GUÍA MAESTRA: Monitoreo y Rendimiento en SQL Server (Semana 13)

Objetivo: Pasar de un enfoque reactivo ("arreglar cuando falla") a uno proactivo ("optimizar el diseño").

FASE 1: DIAGNÓSTICO (¿Qué está pasando?)

Antes de optimizar, necesitamos saber dónde duele. Usamos **DMVs (Dynamic Management Views)** para ver la salud del servidor sin frenarlo (a diferencia del antiguo SQL Profiler).

1.1. Identificar las consultas más pesadas (Top 5 CPU)

Este script te muestra las consultas que históricamente han consumido más procesador.

SQL

```
-- REPORTE 1: Top 5 Consultas por Consumo de CPU
SELECT TOP 5
 qs.total_worker_time AS CPU_Total, -- Tiempo total de
CPU usado
 qs.execution_count AS Veces_Ejecutada, -- Cuántas veces se
corrió
 qs.total_worker_time / qs.execution_count AS
Promedio_CPU_Por_Ejecucion,
 -- Extraer el texto exacto de la consulta
 SUBSTRING(qt.text,qs.statement_start_offset/2 +1,
 (CASE WHEN qs.statement_end_offset = -1
 THEN LEN(CONVERT(nvarchar(max), qt.text)) * 2
 ELSE qs.statement_end_offset end -
qs.statement_start_offset)/2)
 AS Query_Text
FROM sys.dm_exec_query_stats qs
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) qt
ORDER BY qs.total_worker_time DESC;
```

1.2. Detectar "Table Scans" (Lecturas ineficientes)

Este es un script avanzado. Busca consultas que leen muchas páginas de datos (Logical Reads) pero devuelven pocas filas. Esto es el síntoma #1 de que faltan índices.

SQL

```
-- REPORTE 2: Consultas con muchas lecturas (Posibles Table Scans)
SELECT TOP 10
 st.text AS QueryText,

 qs.execution_count,
 qs.total_logical_reads / qs.execution_count AS
```

```
Promedio_Lecturas_Logicas,
 qs.total_rows AS Filas_Devueltas,
 qs.last_worker_time AS Tiempo_CPU
FROM sys.dm_exec_query_stats qs
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) st
-- Ordenar por mayor cantidad de lecturas promedio
ORDER BY (qs.total_logical_reads / qs.execution_count) DESC;
```

FASE 2: OPTIMIZACIÓN (¿Cómo lo mejoramos?)

La optimización tiene dos frentes: **Escribir bien el código (T-SQL)** y **Estructurar bien los datos (Índices)**.

2.1. Optimización de Consultas T-SQL (Concepto SARGable)

Para que SQL Server use un índice, la columna en el `WHERE` debe estar "limpia".
SQL

```
-- EJEMPLO PRÁCTICO: Reescritura de Consultas

-- ❌ MAL (No SARGable):
-- Al aplicar LEFT(), SQL debe calcular la función para CADA fila
de la tabla.
-- Ignorará los índices y hará un Table Scan (lento).
SELECT * FROM Ventas WHERE LEFT(NombreCliente, 4) = 'Juan';

-- ✅ BIEN (SARGable):
-- Usamos LIKE. SQL puede ir directamente al índice y buscar el
rango que empieza con 'Juan'.
SELECT * FROM Ventas WHERE NombreCliente LIKE 'Juan%';

-- ❌ MAL: Conversión Implícita
-- Si 'Codigo' es VARCHAR y comparas con número, SQL convierte toda
la tabla.
SELECT * FROM Productos WHERE CodigoProducto = 100;

-- ✅ BIEN: Tipos de datos correctos
SELECT * FROM Productos WHERE CodigoProducto = '100';
```

2.2. Mantenimiento de Índices (Fragmentación)[1]

Los índices se desordenan con el uso. Este script diagnostica y repara.[2]
SQL

```
-- MANTENIMIENTO 1: Verificar Fragmentación
SELECT
 dbtables.[name] AS 'Tabla',
 dbindexes.[name] AS 'Indice',
 indexstats.avg_fragmentation_in_percent AS
'Fragmentacion_Porcentaje'

FROM sys.dm_db_index_physical_stats(DB_ID(), NULL, NULL, NULL,
NULL) indexstats
INNER JOIN sys.tables dbtables ON dbtables.[object_id] =
indexstats.[object_id]
```

```
INNER JOIN sys.indexes dbindexes ON dbindexes.[object_id] =
indexstats.[object_id]
AND indexstats.index_id = dbindexes.index_id
WHERE indexstats.avg_fragmentation_in_percent > 10; -- Solo mostrar
casos graves

-- MANTENIMIENTO 2: Corrección (Ejemplo de script lógico)
-- Si la fragmentación es alta (>30%), reconstruimos todo el
índice.
ALTER INDEX IX_DetalleVentas_ProductoID ON DetalleVentas REBUILD;

-- Si es media (5-30%), solo reorganizamos las hojas (más ligero).
ALTER INDEX IX_DetalleVentas_ProductoID ON DetalleVentas
REORGANIZE;
```

2.3. Índices Faltantes (Lo que SQL Server "desearía" tener)

SQL Server sabe qué índices le ayudarían a correr más rápido. Este script avanzado consulta esa memoria interna.

SQL

```
-- REPORTE 3: Índices Sugeridos por el Motor
SELECT
 -- Cálculo de impacto: (Búsquedas * Costo * Impacto)
 migs.user_seeks * migs.avg_total_user_cost *
(migs.avg_user_impact / 100.0) AS Puntaje_Mejora,
 mid.statement AS [Tabla],
 mid.equality_columns AS [Columnas_Busqueda_Exacta],
 mid.inequality_columns AS [Columnas_Busqueda_Rango],
 mid.included_columns AS [Columnas_Include]
FROM sys.dm_db_missing_index_group_stats AS migs WITH (NOLOCK)
INNER JOIN sys.dm_db_missing_index_groups AS mig WITH (NOLOCK)
 ON migs.group_handle = mig.index_group_handle
INNER JOIN sys.dm_db_missing_index_details AS mid WITH (NOLOCK)
 ON mig.index_handle = mid.index_handle
ORDER BY Puntaje_Mejora DESC;
```

FASE 3: GESTIÓN Y CONTROL (Prevención de Desastres)

Aquí manejamos la concurrencia (múltiples usuarios a la vez) y limitamos recursos para que nadie tumbe el servidor.

3.1. Detectar Bloqueos en Tiempo Real

Si el sistema se "congela", ejecuta esto para ver quién está bloqueando a quién.

SQL

```
-- MONITOREO: ¿Quién bloquea a quién?
SELECT
 r.session_id AS [ID_Sesion_Victima],
 r.blocking_session_id AS [ID_Sesion_Culpable],
 r.wait_type AS [Tipo_Espera],
```

```
 r.wait_time AS [Tiempo_Espera_ms],
 t.text AS [Query_Que_Se_Esta_Ejecutando]
FROM sys.dm_exec_requests r
CROSS APPLY sys.dm_exec_sql_text(r.sql_handle) t
WHERE r.blocking_session_id <> 0; -- Solo mostrar si hay bloqueo
```

3.2. Resource Governor (Control de Recursos)

Este es el código completo para limitar la CPU a un grupo de usuarios específico (ej. Reportes).

SQL

```
-- CONFIGURACIÓN DE RESOURCE GOVERNOR

-- PASO 1: Crear la Función Clasificadora
-- Esta función decide a qué grupo pertenece cada usuario que se
conecta.
CREATE FUNCTION dbo.ClasificadorUsuarios() RETURNS SYSNAME WITH
SCHEMABINDING
AS
BEGIN
 DECLARE @NombreGrupo SYSNAME;
 -- Si el usuario es 'UsuarioReportes', lo mandamos al grupo
restringido
 IF SUSER_NAME() = 'UsuarioReportes'
 SET @NombreGrupo = 'GrupoReportes';
 ELSE
 SET @NombreGrupo = 'default'; -- El resto va al grupo por
defecto
 RETURN @NombreGrupo;
END;
GO

-- PASO 2: Crear el Pool de Recursos (La restricción física)
-- Aquí decimos: "Este pool nunca usará más del 20% de la CPU si el
servidor está ocupado".
CREATE RESOURCE POOL PoolReportes
WITH (MAX_CPU_PERCENT = 20);
GO

-- PASO 3: Crear el Grupo de Trabajo y unirlo al Pool
CREATE WORKLOAD GROUP GrupoReportes
USING PoolReportes;
GO

-- PASO 4: Activar el Gobernador con la función clasificadora
ALTER RESOURCE GOVERNOR WITH (CLASSIFIER_FUNCTION =
dbo.ClasificadorUsuarios);
ALTER RESOURCE GOVERNOR RECONFIGURE;
GO
```

Conclusión Final de[3]l Análisis

Al unir todos estos conceptos, el ciclo de vida para la **Semana 13** queda así:

1. **Monitoreas** con los scripts de la Fase 1 para encontrar queries lentos.
2. **Analizas** esos queries: ¿Están escritos mal (No SARGable)? ¿Faltan Índices (Missing

Index Script)?
3. **Aplicas la solución:** Reescribes el query o creas el índice y verificas su fragmentación.
4. **Proteges** el servidor configurando alertas de Bloqueos y usando Resource Governor para procesos masivos.

1. Herramientas de Diagnóstico

SQL Profiler (El Veterano)

- **Concepto:** Es una herramienta gráfica de interfaz antigua que permite ver las instrucciones SQL que llegan al servidor en tiempo real.
- **Analogía:** Es como poner una cámara de seguridad en cada pasillo de una tienda. Ves todo, pero la instalación de tantas cámaras hace que la tienda funcione más lento.
- **Estado:** *Obsoleto (Deprecated)*. Microsoft recomienda dejar de usarlo porque consume muchos recursos del servidor.

Extended Events (XEvents - El Estándar)

- **Concepto:** Es el sistema de monitoreo moderno. Es ligero, flexible y está integrado profundamente en el núcleo de SQL Server. Captura eventos y los guarda en un buffer de memoria antes de escribirlos en disco para no afectar el rendimiento.
- **Analogía:** Es como la "Caja Negra" de un avión. Graba silenciosamente miles de parámetros, pero solo te molesta o consume energía si decides abrirla para analizar un accidente.

2. Motor de Almacenamiento (Discos y Estructura)

Estadísticas (Statistics)

- **Concepto:** Son objetos binarios (BLOBs) que contienen histogramas sobre la distribución de los datos en una columna. El optimizador las lee para "adivinar" cuántas filas devolverá una consulta.
- **Analogía:** Un mapa de densidad poblacional. Si vas a buscar a "Juan", la estadística le dice al motor: "En esta zona hay 1 millón de personas, mejor ve en autobús (Scan)" o "Aquí solo hay 3 personas, ve caminando (Seek)".
- **Problema:** Si están desactualizadas, el motor tomará malas decisiones (como ir caminando a buscar entre un millón de personas).

Índice Clustered (Agrupado)

- **Concepto:** Determina el orden físico de los datos en el disco. Las páginas de datos *son* el índice.
- **Regla de Oro:** Solo puede haber **uno** por tabla (usualmente la Llave Primaria).
- **Analogía:** Una guía telefónica (listín). Los datos están ordenados alfabéticamente por apellido. No puedes tener dos órdenes físicos distintos al mismo tiempo.

Índice Non-Clustered (No Agrupado)

- **Concepto:** Es una estructura separada que contiene una copia de las columnas indexadas y un puntero hacia la fila de datos real (en el índice Clustered).
- **Analogía:** El índice al final de un libro de texto. Buscas "Fotosíntesis", te dice "Página 45", y luego vas a la página 45 a leer el contenido completo.

Fragmentación

- **Concepto:** Ocurre cuando el orden lógico de las páginas del índice no coincide con el orden físico en el disco, o cuando hay mucho espacio vacío en las páginas debido a borrados/inserciones.
- **Analogía:** Una biblioteca donde los libros deberían estar ordenados, pero hay estantes medio vacíos y libros de la "A" mezclados en la sección de la "Z". El bibliotecario (SQL Server) tarda más en encontrar lo que busca.

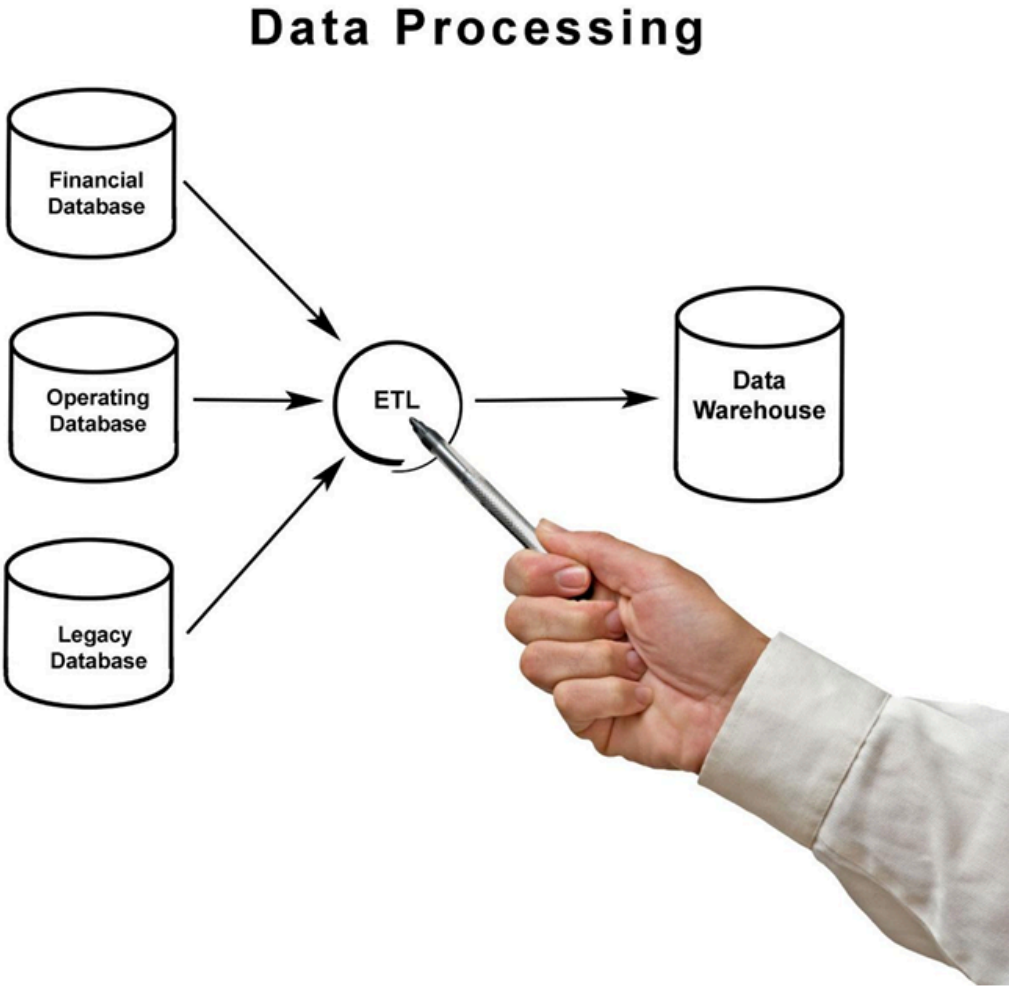
3. Concurrencia (Múltiples usuarios)

Bloqueo (Locking)

- **Concepto:** Es un mecanismo de protección normal. Si una transacción quiere modificar un dato, pone un "candado" para que nadie más lo lea o escriba hasta que termine. Garantiza la consistencia (la "C" de ACID).
- **Analogía:** El probador de una tienda de ropa. Si tú estás dentro, la puerta está cerrada. Los demás deben esperar afuera hasta que salgas.

Deadlock (Abrazo Mortal - Error 1205)

- **Concepto:** Una situación circular donde dos procesos se bloquean mutuamente y ninguno puede avanzar. SQL Server elige a una "víctima", mata su proceso y deja continuar al otro.
- **Analogía:** Un cruce de calles estrecho. El coche A no puede avanzar porque el coche B le bloquea, y el coche B no puede avanzar porque el coche A le bloquea. La grúa (SQL Server) debe quitar uno para que el tráfico fluya.



4. Ejecución y Optimización

Plan de Ejecución

- **Concepto:** Es la hoja de ruta que genera el Optimizador de Consultas. Detalla paso a paso cómo se obtendrán los datos (qué índices usar, cómo unir tablas, cómo ordenar).
- **Analogía:** La ruta que calcula Google Maps antes de que empieces a conducir. Evalúa el tráfico y la distancia para darte la ruta más rápida.

SARGable (Search ARGument ABLE)

- **Concepto:** Se refiere a escribir predicados `WHERE` de forma que el motor pueda aprovechar los índices (Index Seek) en lugar de leer toda la tabla (Index Scan).
- **Regla:** No manipules la columna en la izquierda de la comparación.
  - *No SARGable:* `WHERE YEAR(Fecha) = 2023` (El motor está ciego).
  - *SARGable:* `WHERE Fecha >= '20230101' AND Fecha < '20240101'` (El motor ve el índice).

5. Gobernanza

Resource Governor

- **Concepto:** Una característica de SQL Server Enterprise que permite limitar la cantidad de CPU y Memoria que pueden usar ciertas cargas de trabajo o usuarios específicos.
- **Analogía:** Un sistema de reservación en un restaurante. Tienes mesas reservadas para VIPs (Procesos Críticos) y mesas limitadas para gente que entra sin reserva (Reportes pesados), asegurando que los VIPs siempre tengan servicio rápido sin importar cuán lleno esté el restaurante.

¿Te ayuda este desglose conceptual a entender mejor el código que te pasé antes?