UNIVERSIDAD PERUANA LOS ANDES

"FACULTAD DE INGENIERÍA"

ESCUELA PROFESIONAL "SISTEMAS Y COMPUTACIÓN"

Monitoreo y rendimiento

CURSO: Base de Datos II

PROFESOR: Ing. Fernández Bejarano Raúl Enrique

ESTUDIANTE: Carrillo Cárdenas José

CICLO: V

SECCIÓN: A1

HUANCAYO PERÚ

2025

Semana 13: Monitoreo y rendimiento

Objetivo: Optimizar la eficiencia de SQL Server y prevenir problemas de rendimiento.

Temas:

1. Análisis de rendimiento con SQL Profiler y Extended Events.

SQL Profiler es una herramienta gráfica de Microsoft SQL Server que permite supervisar, rastrear y analizar la actividad de una instancia del motor de base de datos o de Analysis Services. Se utiliza para detectar y diagnosticar problemas de rendimiento, como consultas lentas**,** capturando datos de eventos específicos que luego se pueden guardar y analizar.

Extended Events es un sistema de monitorización de rendimiento ligero y configurable en SQL Server que permite recopilar, supervisar y analizar datos para solucionar problemas y diagnosticar el rendimiento de la base de datos. Sustituye a la herramienta obsoleta SQL Profiler al ofrecer una arquitectura escalable con un impacto mínimo en el rendimiento del sistema.

Crear sesión Extended Events

```
CREATE EVENT SESSION [MonitoreoRendimiento] ON SERVER
ADD EVENT sqlserver.rpc_completed
(
    ACTION (sqlserver.sql_text, sqlserver.client_hostname)
    WHERE (duration > 1000)    -- consultas que tardan más de 1 segundo
),
ADD EVENT sqlserver.sql_batch_completed
(
    ACTION (sqlserver.sql_text, sqlserver.session_id)
    WHERE (duration > 1000)
),
ADD EVENT sqlserver.lock_deadlock,
ADD EVENT sqlserver.lock_deadlock_chain,
ADD EVENT sqlserver.blocked_process_report,
ADD EVENT sqlserver.error_reported
(
    WHERE ([severity] > 15)   -- errores críticos
)
ADD TARGET package0.event_file
(
    SET filename = N'C:\XE\MonitoreoRendimiento.xel',
        max_file_size = 100,
        max_rollover_files = 10
)
WITH
(
    MAX_MEMORY = 4096 KB,
    EVENT_RETENTION_MODE = ALLOW_SINGLE_EVENT_LOSS,
    MAX_DISPATCH_LATENCY = 5 SECONDS,
```

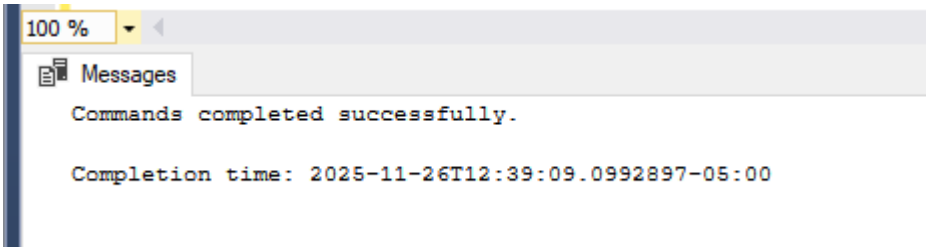
```
    TRACK_CAUSALITY = ON
);
GO
```

Crear nuevamente el target con ruta válida

```
ALTER EVENT SESSION [MonitoreoRendimiento]
ON SERVER
ADD TARGET package0.event_file
(
    SET filename = N'C:\XE\MonitoreoRendimiento.xel',
        max_file_size = 100,
        max_rollover_files = 5
);
GO
```

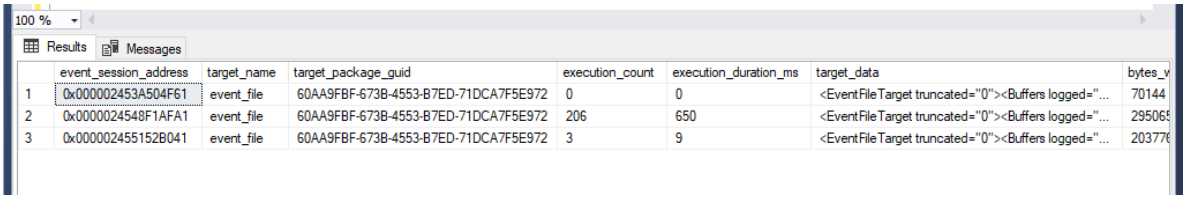
Iniciar la sesión

```
ALTER EVENT SESSION [MonitoreoRendimiento]
ON SERVER
STATE = START;
GO
```



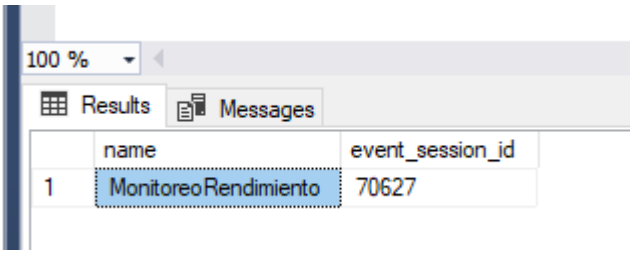
Validar que ya funciona

```
SELECT *
FROM sys.dm_xe_sessions
WHERE name = 'MonitoreoRendimiento';

SELECT *
FROM sys.dm_xe_session_targets
WHERE target_name = 'event_file';
```



Validar que la sesión existe

```
SELECT name, event_session_id
FROM sys.server_event_sessions
WHERE name = 'MonitoreoRendimiento';
```



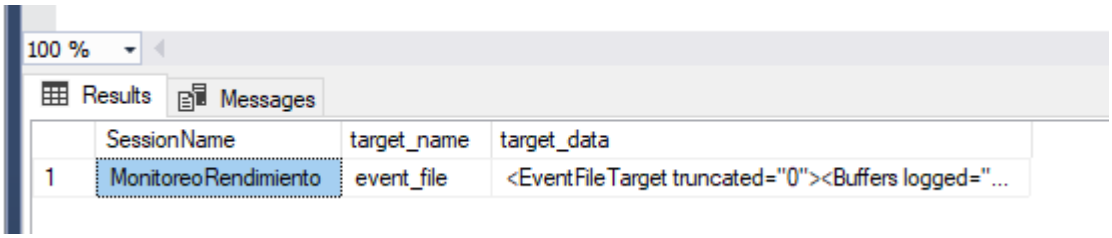
100 %

Results | Messages

	name	event_session_id
1	MonitoreoRendimiento	70627

Validar el target

```
SELECT s.name AS SessionName,
       t.target_name,
       t.target_data
FROM sys.dm_xe_session_targets t
JOIN sys.dm_xe_sessions s
    ON t.event_session_address = s.address
WHERE s.name = 'MonitoreoRendimiento';
```



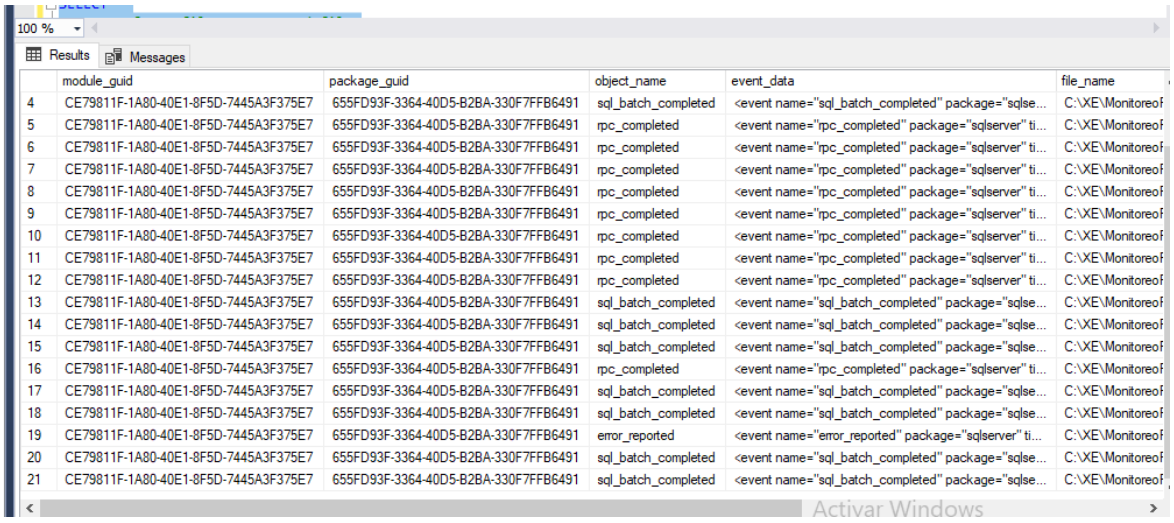
100 %

Results | Messages

	SessionName	target_name	target_data
1	MonitoreoRendimiento	event_file	<EventFileTarget truncated="0"><Buffers logged="...

Leer los eventos capturados

```
SELECT *
FROM sys.fn_xe_file_target_read_file
(
    'C:\XE\MonitoreoRendimiento*.xel',
    NULL,
    NULL,
    NULL
);
```



100 %

Results | Messages

	module_guid	package_guid	object_name	event_data	file_name
4	CE79811F-1A80-40E1-8F5D-7445A3F375E7	655FD93F-3364-40D5-B2BA-330F7FFB6491	sql_batch_completed	<event name="sql_batch_completed" package="sqlse...	C:\XE\Monitoreof
5	CE79811F-1A80-40E1-8F5D-7445A3F375E7	655FD93F-3364-40D5-B2BA-330F7FFB6491	rpc_completed	<event name="rpc_completed" package="sqlserver" ti...	C:\XE\Monitoreof
6	CE79811F-1A80-40E1-8F5D-7445A3F375E7	655FD93F-3364-40D5-B2BA-330F7FFB6491	rpc_completed	<event name="rpc_completed" package="sqlserver" ti...	C:\XE\Monitoreof
7	CE79811F-1A80-40E1-8F5D-7445A3F375E7	655FD93F-3364-40D5-B2BA-330F7FFB6491	rpc_completed	<event name="rpc_completed" package="sqlserver" ti...	C:\XE\Monitoreof
8	CE79811F-1A80-40E1-8F5D-7445A3F375E7	655FD93F-3364-40D5-B2BA-330F7FFB6491	rpc_completed	<event name="rpc_completed" package="sqlserver" ti...	C:\XE\Monitoreof
9	CE79811F-1A80-40E1-8F5D-7445A3F375E7	655FD93F-3364-40D5-B2BA-330F7FFB6491	rpc_completed	<event name="rpc_completed" package="sqlserver" ti...	C:\XE\Monitoreof
10	CE79811F-1A80-40E1-8F5D-7445A3F375E7	655FD93F-3364-40D5-B2BA-330F7FFB6491	rpc_completed	<event name="rpc_completed" package="sqlserver" ti...	C:\XE\Monitoreof
11	CE79811F-1A80-40E1-8F5D-7445A3F375E7	655FD93F-3364-40D5-B2BA-330F7FFB6491	rpc_completed	<event name="rpc_completed" package="sqlserver" ti...	C:\XE\Monitoreof
12	CE79811F-1A80-40E1-8F5D-7445A3F375E7	655FD93F-3364-40D5-B2BA-330F7FFB6491	rpc_completed	<event name="rpc_completed" package="sqlserver" ti...	C:\XE\Monitoreof
13	CE79811F-1A80-40E1-8F5D-7445A3F375E7	655FD93F-3364-40D5-B2BA-330F7FFB6491	sql_batch_completed	<event name="sql_batch_completed" package="sqlse...	C:\XE\Monitoreof
14	CE79811F-1A80-40E1-8F5D-7445A3F375E7	655FD93F-3364-40D5-B2BA-330F7FFB6491	sql_batch_completed	<event name="sql_batch_completed" package="sqlse...	C:\XE\Monitoreof
15	CE79811F-1A80-40E1-8F5D-7445A3F375E7	655FD93F-3364-40D5-B2BA-330F7FFB6491	sql_batch_completed	<event name="sql_batch_completed" package="sqlse...	C:\XE\Monitoreof
16	CE79811F-1A80-40E1-8F5D-7445A3F375E7	655FD93F-3364-40D5-B2BA-330F7FFB6491	rpc_completed	<event name="rpc_completed" package="sqlserver" ti...	C:\XE\Monitoreof
17	CE79811F-1A80-40E1-8F5D-7445A3F375E7	655FD93F-3364-40D5-B2BA-330F7FFB6491	sql_batch_completed	<event name="sql_batch_completed" package="sqlse...	C:\XE\Monitoreof
18	CE79811F-1A80-40E1-8F5D-7445A3F375E7	655FD93F-3364-40D5-B2BA-330F7FFB6491	sql_batch_completed	<event name="sql_batch_completed" package="sqlse...	C:\XE\Monitoreof
19	CE79811F-1A80-40E1-8F5D-7445A3F375E7	655FD93F-3364-40D5-B2BA-330F7FFB6491	error_reported	<event name="error_reported" package="sqlserver" ti...	C:\XE\Monitoreof
20	CE79811F-1A80-40E1-8F5D-7445A3F375E7	655FD93F-3364-40D5-B2BA-330F7FFB6491	sql_batch_completed	<event name="sql_batch_completed" package="sqlse...	C:\XE\Monitoreof
21	CE79811F-1A80-40E1-8F5D-7445A3F375E7	655FD93F-3364-40D5-B2BA-330F7FFB6491	sql_batch_completed	<event name="sql_batch_completed" package="sqlse...	C:\XE\Monitoreof

Activar Windows

JUSTIFICACIÓN TÉCNICA

==========

✓ Extended Events es la herramienta moderna
SQL Profiler está en desuso (deprecated).
Extended Events es:
Más liviano
Consume menos recursos
Permite capturar más eventos
Escalable en producción
Permite filtrar eventos para evitar sobrecarga
✓ Los eventos configurados cubren las principales causas de problemas
sql_batch_completed / rpc_completed → detectan consultas lentas
lock_deadlock / deadlock_chain → detectan deadlocks
blocked_process_report → analiza bloqueos y esperas
error_reported → registra errores graves (priority > 15)

BUENAS PRÁCTICAS UTILIZADAS

==========

📌 1. Nunca usar SQL Profiler en producción
Profiler captura todo → esto causa:
Caída de rendimiento
CPU alta
Bloqueo de procesos
Contención en el buffer
Por eso se usa solo en desarrollo, nunca en producción.
Extended Events es la alternativa actual y recomendada.

Semana 13: Monitoreo y rendimiento
Objetivo: Optimizar la eficiencia de SQL Server y prevenir problemas de rendimiento.
-Estadísticas e índices (creación, fragmentación, mantenimiento).

Los **índices** aceleran la recuperación de datos en SQL Server, mientras que las **estadísticas** informan al optimizador de consultas sobre los datos para elegir el mejor plan de ejecución. La **creación** de índices optimiza las búsquedas, la **fragmentación** (desorganización física o lógica) puede causar lentitud y el **mantenimiento** regular (reconstrucción y reorganización de índices, y actualización de estadísticas) es crucial para mantener un rendimiento óptimo.

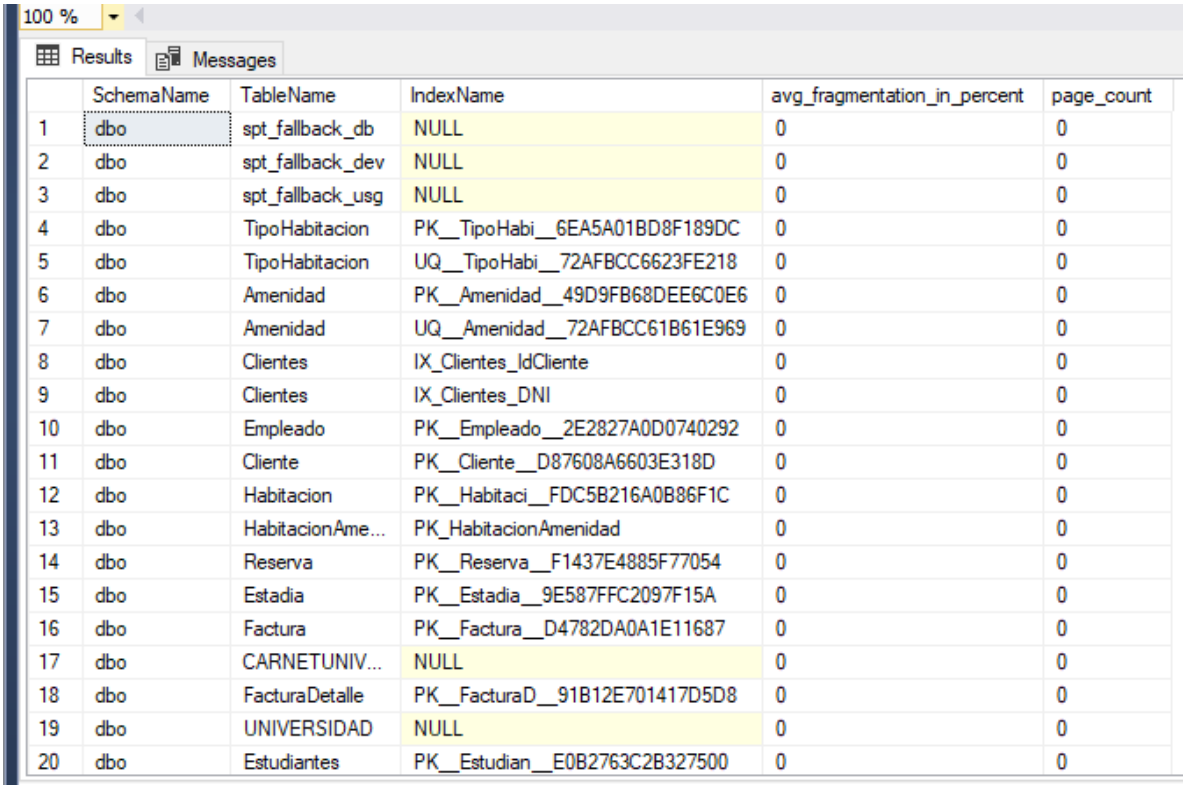
Script de la solución en T SQL

Crear un índice

```
CREATE NONCLUSTERED INDEX IX_Clientes_DNI
ON dbo.Clientes (DNI)
INCLUDE (Nombre, Apellido);
```

Ver el nivel de fragmentación de índices

```
SELECT
    dbschemas.[name] AS SchemaName,
    dbtables.[name] AS TableName,
    dbindexes.[name] AS IndexName,
    indexstats.avg_fragmentation_in_percent,
    indexstats.page_count
FROM sys.dm_db_index_physical_stats(DB_ID(), NULL, NULL, NULL, 'DETAILED')
indexstats
JOIN sys.tables dbtables ON dbtables.[object_id] = indexstats.[object_id]
JOIN sys.schemas dbschemas ON dbschemas.[schema_id] = dbtables.[schema_id]
JOIN sys.indexes dbindexes ON dbindexes.[object_id] = indexstats.[object_id]
                           AND dbindexes.index_id = indexstats.index_id
WHERE indexstats.database_id = DB_ID()
ORDER BY indexstats.avg_fragmentation_in_percent DESC;
```



100 %

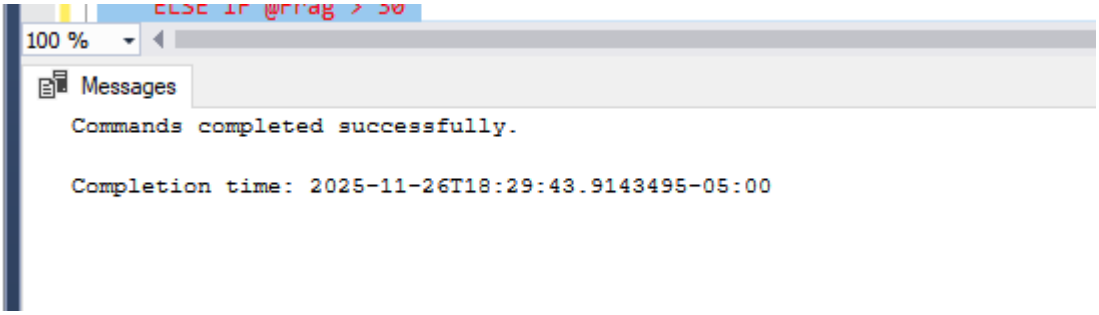
Results | Messages

	SchemaName	TableName	IndexName	avg_fragmentation_in_percent	page_count
1	dbo	spt_fallback_db	NULL	0	0
2	dbo	spt_fallback_dev	NULL	0	0
3	dbo	spt_fallback_usg	NULL	0	0
4	dbo	TipoHabitacion	PK__TipoHabi__6EA5A01BD8F189DC	0	0
5	dbo	TipoHabitacion	UQ__TipoHabi__72AFBCC6623FE218	0	0
6	dbo	Amenidad	PK__Amenidad__49D9FB68DEE6C0E6	0	0
7	dbo	Amenidad	UQ__Amenidad__72AFBCC61B61E969	0	0
8	dbo	Clientes	IX_Clientes_IdCliente	0	0
9	dbo	Clientes	IX_Clientes_DNI	0	0
10	dbo	Empleado	PK__Empleado__2E2827A0D0740292	0	0
11	dbo	Cliente	PK__Cliente__D87608A6603E318D	0	0
12	dbo	Habitacion	PK__Habitaci__FDC5B216A0B86F1C	0	0
13	dbo	HabitacionAme...	PK_HabitacionAmenidad	0	0
14	dbo	Reserva	PK__Reserva__F1437E4885F77054	0	0
15	dbo	Estadia	PK__Estadia__9E587FFC2097F15A	0	0
16	dbo	Factura	PK__Factura__D4782DA0A1E11687	0	0
17	dbo	CARNETUNIV...	NULL	0	0
18	dbo	FacturaDetalle	PK__FacturaD__91B12E701417D5D8	0	0
19	dbo	UNIVERSIDAD	NULL	0	0
20	dbo	Estudiantes	PK__Estudian__E0B2763C2B327500	0	0

Mantenimiento automático de índices (REORGANIZE / REBUILD)

```
DECLARE @TableName NVARCHAR(200);
DECLARE @IndexName NVARCHAR(200);
DECLARE @SQL NVARCHAR(MAX);

DECLARE cur CURSOR FOR
SELECT
```

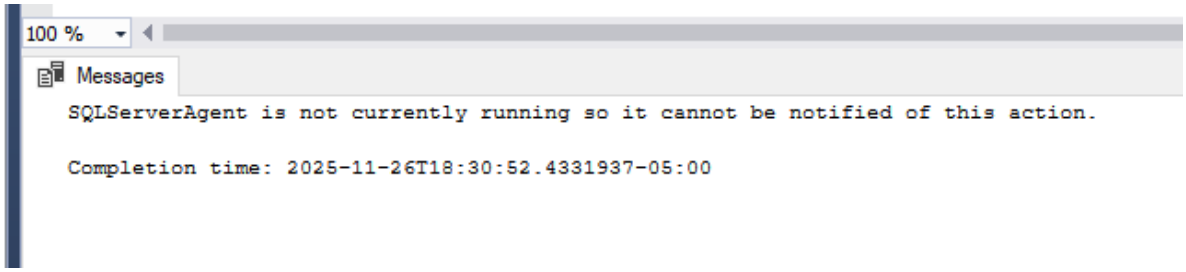
```
    t.name AS TableName,
    i.name AS IndexName
FROM sys.dm_db_index_physical_stats(DB_ID(), NULL, NULL, NULL, 'LIMITED') d
JOIN sys.indexes i ON d.object_id = i.object_id AND d.index_id = i.index_id
JOIN sys.tables t ON t.object_id = d.object_id
WHERE d.index_id > 0 AND d.page_count > 100; -- evita índices pequeños

OPEN cur;
FETCH NEXT FROM cur INTO @TableName, @IndexName;

WHILE @@FETCH_STATUS = 0
BEGIN
    SET @SQL = '
    DECLARE @Frag FLOAT;
    SELECT @Frag = avg_fragmentation_in_percent
    FROM sys.dm_db_index_physical_stats(
            DB_ID(), OBJECT_ID(''' + @TableName + '''),
            (SELECT index_id FROM sys.indexes WHERE name = ''' + @IndexName + '''
AND object_id = OBJECT_ID(''' + @TableName + ''')),
            NULL, ''LIMITED'');

    IF @Frag BETWEEN 5 AND 30
        ALTER INDEX [' + @IndexName + '] ON [' + @TableName + '] REORGANIZE;
    ELSE IF @Frag > 30
        ALTER INDEX [' + @IndexName + '] ON [' + @TableName + '] REBUILD WITH
(ONLINE = ON);';

    EXEC(@SQL);

    FETCH NEXT FROM cur INTO @TableName, @IndexName;
END

CLOSE cur;
DEALLOCATE cur;
```



Crear un Job para mantenimiento diario

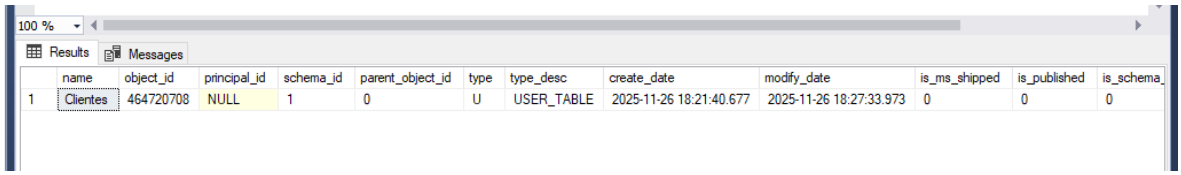
```
EXEC msdb.dbo.sp_add_job
    @job_name = N'Mantenimiento_Indices';

EXEC msdb.dbo.sp_add_jobstep
    @job_name = N'Mantenimiento_Indices',
    @step_name = N'Rebuild_Reorganize',
    @database_name = 'QhatuPeru',
    @command = N'EXEC dbo.usp_MantenimientoIndices;';

EXEC msdb.dbo.sp_add_schedule
    @schedule_name = N'Diario_3AM',
```

```
    @freq_type = 4,           -- diario
    @freq_interval = 1,
    @active_start_time = 030000;

EXEC msdb.dbo.sp_attach_schedule
    @job_name = N'Mantenimiento_Indices',
    @schedule_name = N'Diario_3AM';

EXEC msdb.dbo.sp_add_jobserver
    @job_name = N'Mantenimiento_Indices';
```



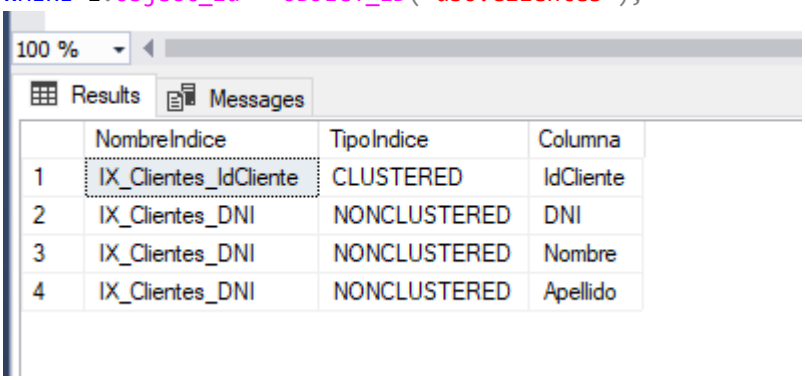
Validar´

```
SELECT *
FROM sys.tables
WHERE name = 'Clientes';
```



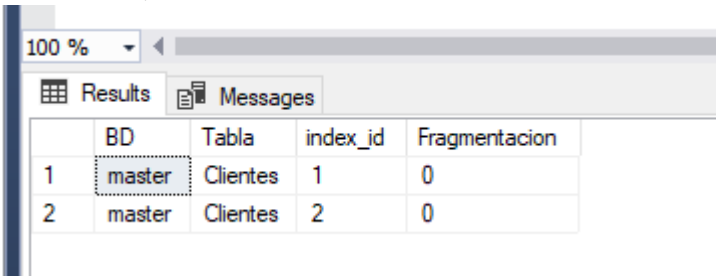
Validar qué índices tiene actualmente la tabla

```
SELECT
    i.name AS NombreIndice,
    i.type_desc AS TipoIndice,
    c.name AS Columna
FROM sys.indexes i
INNER JOIN sys.index_columns ic ON i.object_id = ic.object_id AND i.index_id =
ic.index_id
INNER JOIN sys.columns c ON ic.object_id = c.object_id AND ic.column_id =
c.column_id
WHERE i.object_id = OBJECT_ID('dbo.Clientes');
```



Validar fragmentación de índices

```
SELECT
    DB_NAME(database_id) AS BD,
    OBJECT_NAME(object_id) AS Tabla,
    index_id,
    avg_fragmentation_in_percent AS Fragmentacion
FROM sys.dm_db_index_physical_stats(DB_ID(), OBJECT_ID('dbo.Clientes'), NULL, NULL,
'DETAILED');
```



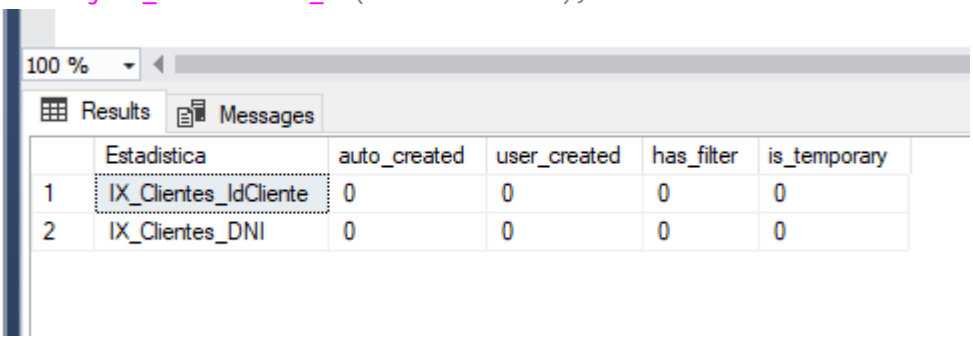
100 %

Results | Messages

	BD	Tabla	index_id	Fragmentacion
1	master	Clientes	1	0
2	master	Clientes	2	0

Validar estadísticas

```
SELECT
    name AS Estadistica,
    auto_created,
    user_created,
    has_filter,
    is_temporary
FROM sys.stats
WHERE object_id = OBJECT_ID('dbo.Clientes');
```



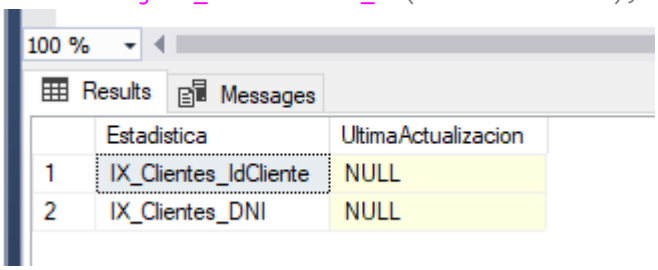
100 %

Results | Messages

	Estadistica	auto_created	user_created	has_filter	is_temporary
1	IX_Clientes_IdCliente	0	0	0	0
2	IX_Clientes_DNI	0	0	0	0

Validar mantenimiento ejecutado

```
SELECT
    s.name AS Estadistica,
    STATS_DATE(s.object_id, s.stats_id) AS UltimaActualizacion
FROM sys.stats s
WHERE s.object_id = OBJECT_ID('dbo.Clientes');
```



100 %

Results | Messages

	Estadistica	UltimaActualizacion
1	IX_Clientes_IdCliente	NULL
2	IX_Clientes_DNI	NULL

JUSTIFICACIÓN TÉCNICA

Un índice clustered organiza físicamente los datos → mejora búsquedas por clave primaria.
Índices no clustered aceleran búsquedas específicas sin afectar el orden físico.
Fragmentación de índices provoca más lecturas de disco y deteriora el rendimiento.
REORGANIZE se usa para fragmentación leve (5–30%) porque es rápido y ONLINE.
REBUILD se usa para fragmentación severa (>30%) porque recrea el índice desde cero y optimiza al máximo.
Actualización de estadísticas mejora los planes de ejecución del optimizador de consultas.
Automatizar un job garantiza mantenimiento continuo sin intervención manual.

✅ 3. BUENAS PRÁCTICAS APLICADAS

✓ 1. No reconstruir índices pequeños
Los índices con menos de 100 páginas no requieren mantenimiento (Microsoft recomienda evitarlo).
✓ 2. Usar ONLINE = ON en REBUILD
Previene bloqueos y evita detener operaciones en la base de datos.
✓ 3. Separar REBUILD y REORGANIZE
Cada uno se usa en su rango óptimo de fragmentación.
✓ 4. Actualizar estadísticas después del mantenimiento
Para que el optimizador use datos recientes.
✓ 5. Crear jobs de mantenimiento fuera de horario laboral
Generalmente 2 AM – 4 AM.
✓ 6. Guardar scripts en procedimientos almacenados
Evitando copiar/pegar código directamente.
✓ 7. Ejecutar con permisos mínimos requeridos
Para evitar riesgos en producción.

Semana 13: Monitoreo y rendimiento
Objetivo: Optimizar la eficiencia de SQL Server y prevenir problemas de rendimiento.
Temas: Administración de transacciones y bloqueos.

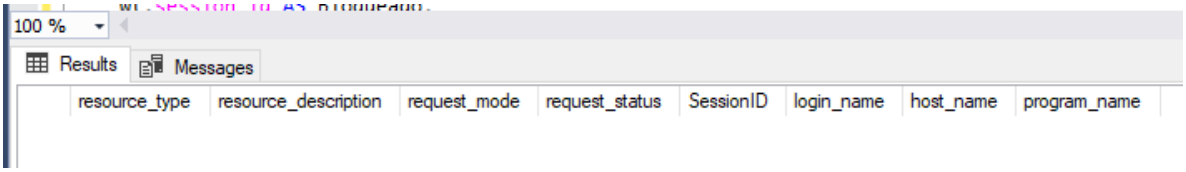
La administración de transacciones y bloqueos en SQL es el proceso de **controlar la concurrencia** y asegurar la **integridad de los datos** en bases de datos con múltiples usuarios. Las transacciones agrupan operaciones SQL como una unidad atómica (todo o nada), mientras que los bloqueos son mecanismos para controlar el acceso a los datos durante la ejecución de estas transacciones. Los

bloqueos **compartidos** permiten lecturas simultáneas, mientras que los **exclusivos** impiden que otros usuarios modifiquen un recurso mientras se está trabajando en él.

Script en T-SQL

Ver bloqueos actuales

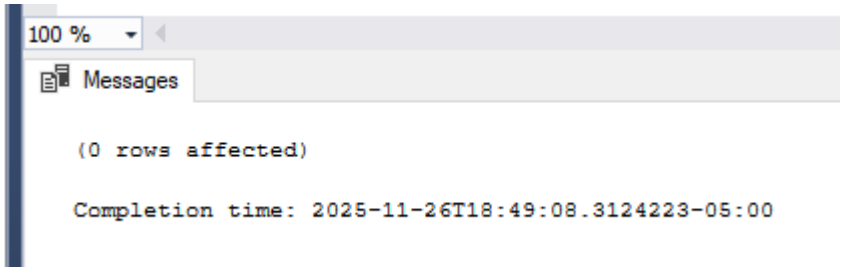
```
-- Mostrar bloqueos activos
SELECT
    tl.resource_type,
    tl.resource_description,
    tl.request_mode,
    tl.request_status,
    tl.request_session_id AS SessionID,
    s.login_name,
    s.host_name,
    s.program_name
FROM sys.dm_tran_locks tl
INNER JOIN sys.dm_exec_sessions s
    ON tl.request_session_id = s.session_id;
```



Simular un bloqueo

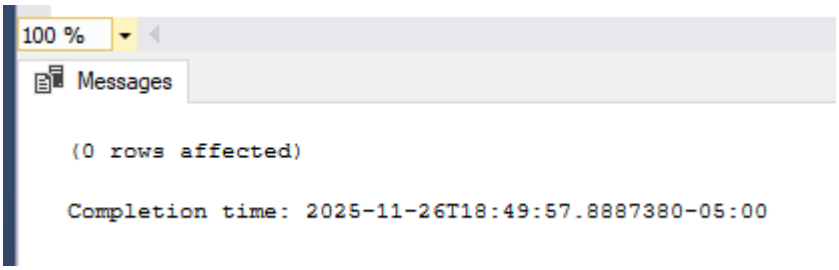
Sesión 1

```
BEGIN TRAN;
UPDATE dbo.Clientes
SET Nombre = 'TEST'
WHERE IdCliente = 1;
-- TRAN abierta → genera bloqueo
```



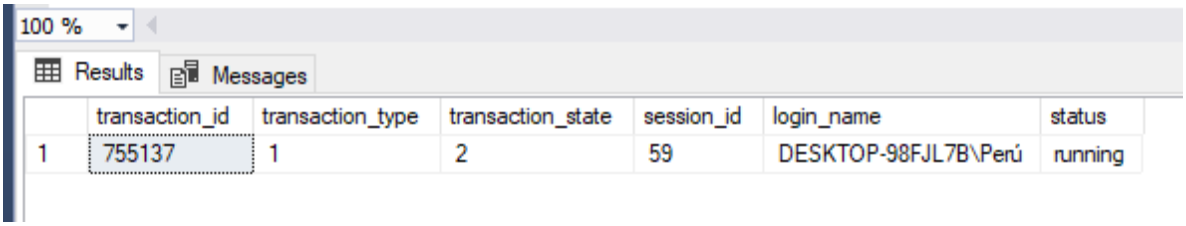
Sesión 2

```
UPDATE dbo.Clientes
SET Nombre = 'OTRO'
WHERE IdCliente = 1;
```



Ver transacciones abiertas

```
SELECT
    at.transaction_id,
    at.transaction_type,
    at.transaction_state,
    s.session_id,
    s.login_name,
    s.status
FROM sys.dm_tran_active_transactions at
JOIN sys.dm_tran_session_transactions st
    ON at.transaction_id = st.transaction_id
JOIN sys.dm_exec_sessions s
    ON s.session_id = st.session_id;
```

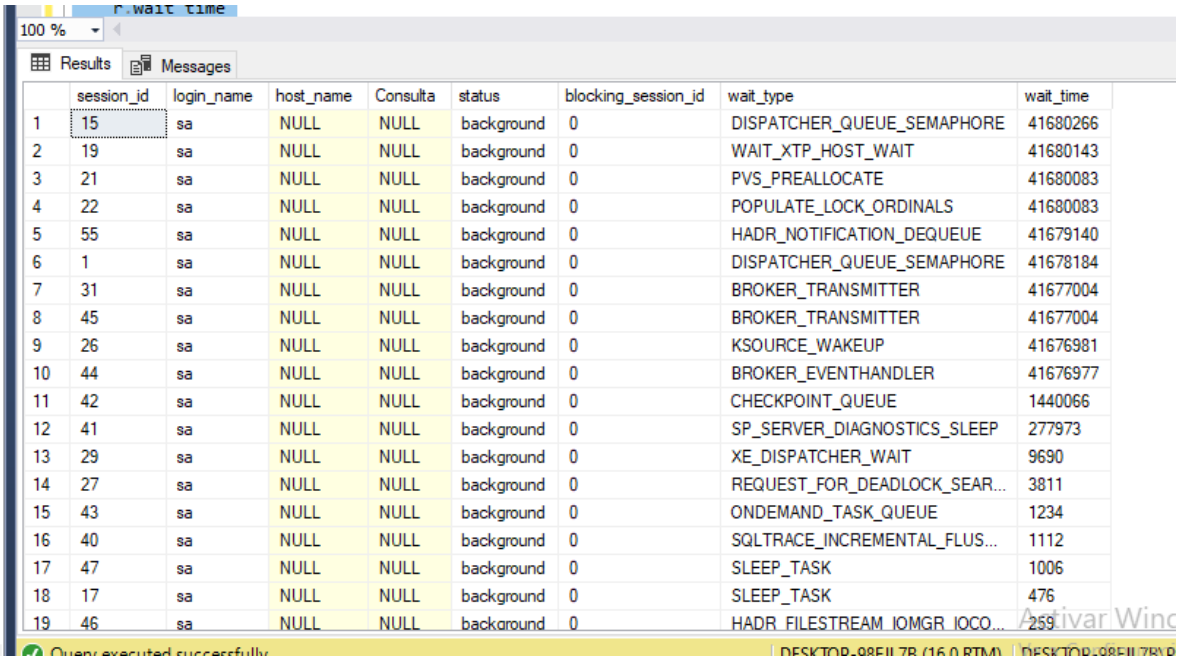


	transaction_id	transaction_type	transaction_state	session_id	login_name	status
1	755137	1	2	59	DESKTOP-98FJL7B\Perú	running

VALIDAR

Consultar TODAS las sesiones bloqueadoras

```
SELECT
    s.session_id,
    s.login_name,
    s.host_name,
    t.text AS Consulta,
    r.status,
    r.blocking_session_id,
    r.wait_type,
    r.wait_time
FROM sys.dm_exec_sessions s
LEFT JOIN sys.dm_exec_requests r ON s.session_id = r.session_id
OUTER APPLY sys.dm_exec_sql_text(r.sql_handle) t
ORDER BY r.wait_time DESC;
```



100 %

Results | Messages

	session_id	login_name	host_name	Consulta	status	blocking_session_id	wait_type	wait_time
1	15	sa	NULL	NULL	background	0	DISPATCHER_QUEUE_SEMAPHORE	41680266
2	19	sa	NULL	NULL	background	0	WAIT_XTP_HOST_WAIT	41680143
3	21	sa	NULL	NULL	background	0	PVS_PREALLOCATE	41680083
4	22	sa	NULL	NULL	background	0	POPULATE_LOCK_ORDINALS	41680083
5	55	sa	NULL	NULL	background	0	HADR_NOTIFICATION_DEQUEUE	41679140
6	1	sa	NULL	NULL	background	0	DISPATCHER_QUEUE_SEMAPHORE	41678184
7	31	sa	NULL	NULL	background	0	BROKER_TRANSMITTER	41677004
8	45	sa	NULL	NULL	background	0	BROKER_TRANSMITTER	41677004
9	26	sa	NULL	NULL	background	0	KSOURCE_WAKEUP	41676981
10	44	sa	NULL	NULL	background	0	BROKER_EVENTHANDLER	41676977
11	42	sa	NULL	NULL	background	0	CHECKPOINT_QUEUE	1440066
12	41	sa	NULL	NULL	background	0	SP_SERVER_DIAGNOSTICS_SLEEP	277973
13	29	sa	NULL	NULL	background	0	XE_DISPATCHER_WAIT	9690
14	27	sa	NULL	NULL	background	0	REQUEST_FOR_DEADLOCK_SEAR...	3811
15	43	sa	NULL	NULL	background	0	ONDEMAND_TASK_QUEUE	1234
16	40	sa	NULL	NULL	background	0	SQLTRACE_INCREMENTAL_FLUS...	1112
17	47	sa	NULL	NULL	background	0	SLEEP_TASK	1006
18	17	sa	NULL	NULL	background	0	SLEEP_TASK	476
19	46	sa	NULL	NULL	background	0	HADR_FILESTREAM_IOMGR_IOCO...	259

Query executed successfully.

JUSTIFICACIÓN TÉCNICA DE LA SOLUCIÓN

La administración de transacciones y bloqueos es crítica para evitar:

✓ Contención de recursos

Cuando muchas transacciones compiten por filas, páginas o tablas.

✓ Deadlocks

Situaciones donde dos transacciones se bloquean mutuamente.

✓ Degradación del rendimiento

Consultas lentas, tiempos de espera, bloqueos prolongados.

Por eso se utilizan:

DMVs como dm_tran_locks, dm_exec_requests y dm_os_waiting_tasks, que permiten ver en tiempo real:

qué sesión bloquea

qué recurso está bloqueado

qué tipo de lock está aplicado

Transacciones explícitas (BEGIN TRAN) para simular y detectar bloqueos.

KILL para liberar recursos cuando una transacción queda colgada.

Estas herramientas permiten a un DBA:

Diagnosticar problemas de concurrencia

Mantener el rendimiento del sistema

Prevenir interrupciones en ambientes productivos

✓ 3. BUENAS PRÁCTICAS UTILIZADAS

1 Usar transacciones cortas
Evitar transacciones largas porque:
generan bloqueos prolongados
afectan rendimiento
pueden causar deadlocks

---

2 Confirmar (COMMIT) o revertir (ROLLBACK) siempre
Nunca dejar transacciones abiertas accidentalmente.

---

3 Usar índices adecuados para reducir bloqueos
Las búsquedas basadas en índices evitan escaneos grandes → menos recursos bloqueados.

---

4 Supervisar continuamente con DMVs
DMVs son herramientas nativas de SQL Server para monitoreo en tiempo real.

---

5 Implementar niveles de aislamiento adecuados
Ejemplo:
READ COMMITTED para la mayoría
READ COMMITTED SNAPSHOT reduce bloqueos
SNAPSHOT evita esperas por lectura

---

6 Evitar actualización masiva sin control
Siempre paginar, usar lotes o TOP (1000) en operaciones grandes.

---

7 Registrar bloqueos recurrentes
Usar Extended Events o Alertas de Deadlocks para diagnóstico continuo.

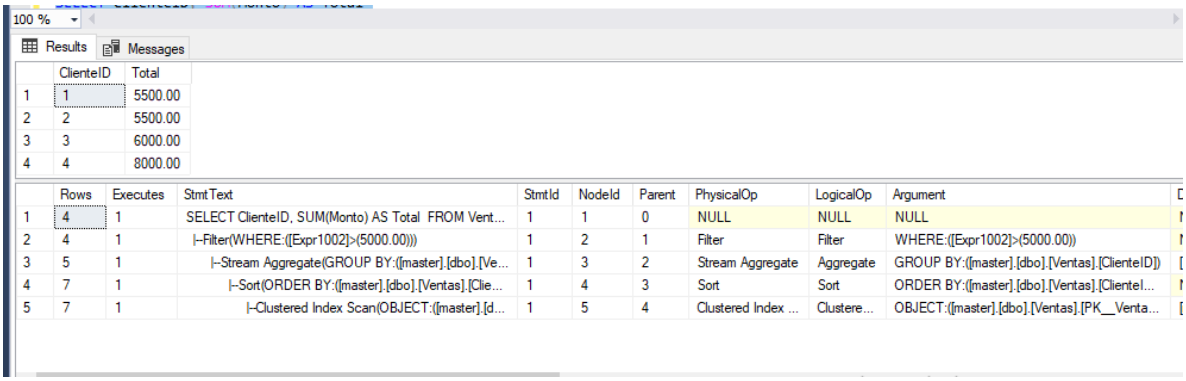
Semana 13: Monitoreo y rendimiento
Objetivo: Optimizar la eficiencia de SQL Server y prevenir problemas de rendimiento.
Temas: Análisis de planes de ejecución.

El análisis de planes de ejecución en SQL es el proceso de **interpretar la estrategia que la base de datos utiliza para ejecutar una consulta**, identificando los pasos, operadores y recursos empleados para recuperar datos. Esta interpretación permite a los desarrolladores y administradores de bases de datos **identificar cuellos de botella, ineficiencias** (como un uso pobre de índices o uniones costosas) y **optimizar las consultas** para mejorar el rendimiento y el uso de recursos.

Script T-SQL

Ver plan de ejecución real

```
SET STATISTICS PROFILE ON;
SET STATISTICS IO ON;
SET STATISTICS TIME ON;
GO

SELECT ClienteID, SUM(Monto) AS Total
FROM Ventas
GROUP BY ClienteID
HAVING SUM(Monto) > 5000;
GO

SET STATISTICS PROFILE OFF;
SET STATISTICS IO OFF;
SET STATISTICS TIME OFF;
GO
```



100 %

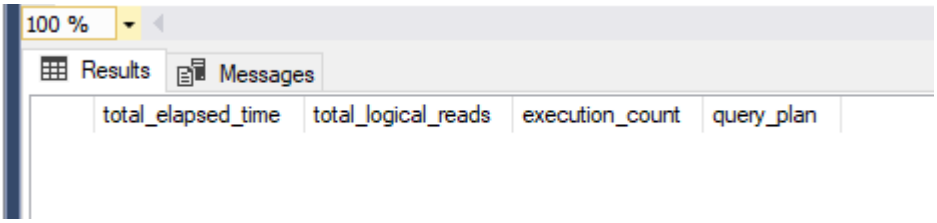
Results | Messages

	ClienteID	Total
1	1	5500.00
2	2	5500.00
3	3	6000.00
4	4	8000.00

	Rows	Executes	StmtText	StmtId	NodeId	Parent	PhysicalOp	LogicalOp	Argument
1	4	1	SELECT ClienteID, SUM(Monto) AS Total FROM Vent...	1	1	0	NULL	NULL	NULL
2	4	1	|--Filter(WHERE:([Expr1002]>(5000.00)))	1	2	1	Filter	Filter	WHERE:([Expr1002]>(5000.00))
3	5	1	|--Stream Aggregate(GROUP BY:([master].[dbo].[Ve...	1	3	2	Stream Aggregate	Aggregate	GROUP BY:([master].[dbo].[Ventas].[ClienteID])
4	7	1	|--Sort(ORDER BY:([master].[dbo].[Ventas].[Clie...	1	4	3	Sort	Sort	ORDER BY:([master].[dbo].[Ventas].[ClienteI...
5	7	1	|--Clustered Index Scan(OBJECT:([master].[d...	1	5	4	Clustered Index ...	Clustere...	OBJECT:([master].[dbo].[Ventas].[PK__Venta...

Consultar planes guardados en caché

```
SELECT
    qs.total_elapsed_time,
    qs.total_logical_reads,
    qs.execution_count,
    qp.query_plan
FROM sys.dm_exec_query_stats qs
CROSS APPLY sys.dm_exec_query_plan(qs.plan_handle) qp
ORDER BY qs.total_logical_reads DESC;
```



100 %

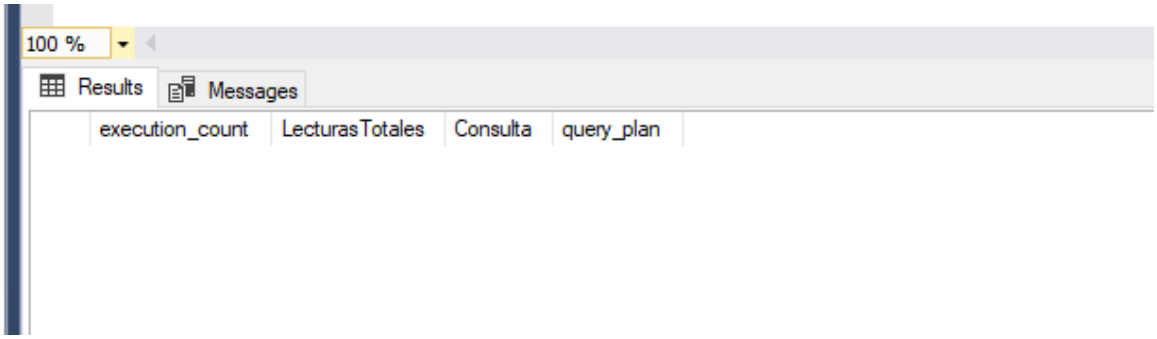
Results | Messages

total_elapsed_time	total_logical_reads	execution_count	query_plan

Buscar consultas con escaneos (SCANS) costosos

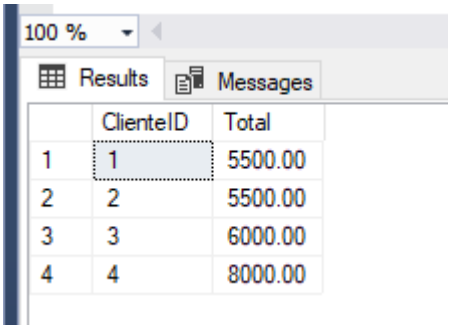
```
SELECT
    qs.execution_count,
    qs.total_logical_reads AS LecturasTotales,
    SUBSTRING(st.text, (qs.statement_start_offset/2)+1,
        ((CASE qs.statement_end_offset
            WHEN -1 THEN DATALENGTH(st.text)
```

```
            ELSE qs.statement_end_offset END
          - qs.statement_start_offset)/2) + 1) AS Consulta,
    qp.query_plan
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
WHERE qp.query_plan.value('declare default element namespace
 "http://schemas.microsoft.com/sqlserver/2004/07/showplan";

(/ShowPlanXML/BatchSequence/Batch/Statements/StmtSimple/QueryPlan/RelOp/@PhysicalOp)
[1]',
 'varchar(100)') IN ('Table Scan','Index Scan')
ORDER BY qs.total_logical_reads DESC;
```



VALIDAR

```
SELECT ClienteID, SUM(Monto) AS Total
FROM Ventas
GROUP BY ClienteID
HAVING SUM(Monto) > 5000;
GO
```



	ClienteID	Total
1	1	5500.00
2	2	5500.00
3	3	6000.00
4	4	8000.00

JUSTIFICACIÓN TÉCNICA

SET SHOWPLAN_XML

Permite ver cómo SQL Server planea ejecutar la consulta sin ejecutarla. Esto ayuda a detectar:

Escaneos de tabla

Mal uso de índices

Falta de estadísticas actualizadas

STATISTICS IO/TIME/PROFILE

Ofrece información detallada del rendimiento:

Lecturas lógicas = indicador fuerte de eficiencia
Tiempo CPU
Operadores utilizados
Costos internos del plan
Consultas a DMVs (Dynamic Management Views)
Permiten identificar:
Consultas más pesadas
Planes de ejecución almacenados
Cuellos de botella
Detección de SCANS
Permite identificar cuándo SQL Server está recorriendo toda una tabla en lugar de usar índices → impacto directo en el rendimiento.

✅ 3. BUENAS PRÁCTICAS

✓ Actualizar estadísticas
UPDATE STATISTICS dbo.Ventas WITH FULLSCAN;
✓ Crear índices basados en los resultados del plan
Evitar TABLE SCAN: agregar índices adecuados
Evitar KEY LOOKUP: usar INCLUDE
✓ Evitar SELECT *
Esto fuerza al optimizador a leer más columnas de las necesarias.
✓ Revisar planes después de cada cambio importante
Cambios de estructura
Cambios de índices
Incremento de volumen de datos
✓ Monitorear continuamente las consultas más pesadas
Utilizar DMVs regularmente para detectar consultas problemáticas.
✓ Evitar cursors y optar por operaciones set-based
Los cursores tienden a generar planes más costosos.
✓ Mantener la fragmentación de índices controlada
Reorganizar cuando fragmentación > 10%
Rebuild cuando > 30%

Semana 13: Monitoreo y rendimiento
Objetivo: Optimizar la eficiencia de SQL Server y prevenir problemas de rendimiento.
Temas: Optimización de consultas T-SQL.

La optimización de consultas T-SQL es el proceso de mejorar la eficiencia de las sentencias de Transact-SQL para que se ejecuten más rápido, usen menos recursos y devuelvan resultados de manera más efectiva. Esto se logra a través de técnicas como el uso adecuado de índices, un filtrado eficiente, la elección correcta de las uniones y la reescritura lógica de la consulta.

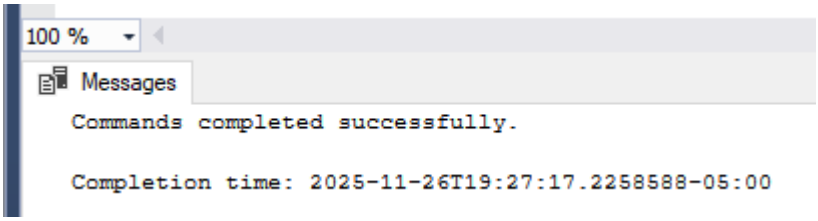
SCRIPT EN T-SQL

Crear tabla de ejemplo (si no existe)

```
IF OBJECT_ID('dbo.Ventas', 'U') IS NOT NULL
    DROP TABLE dbo.Ventas;
GO

CREATE TABLE dbo.Ventas (
    VentaID INT IDENTITY PRIMARY KEY,
    ClienteID INT NOT NULL,
    Fecha DATE NOT NULL,
    Monto DECIMAL(10,2) NOT NULL
);
GO
```

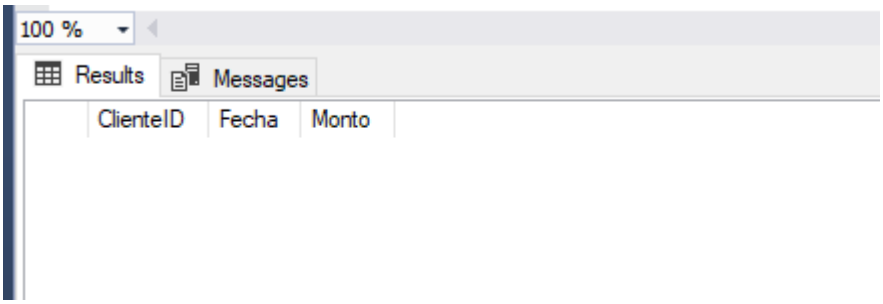
Crear índices sugeridos

```
-- Índice para búsquedas por cliente y rangos de fecha
CREATE NONCLUSTERED INDEX IX_Ventas_Cliente_Fecha
    ON dbo.Ventas (ClienteID, Fecha)
    INCLUDE (Monto);
GO
```



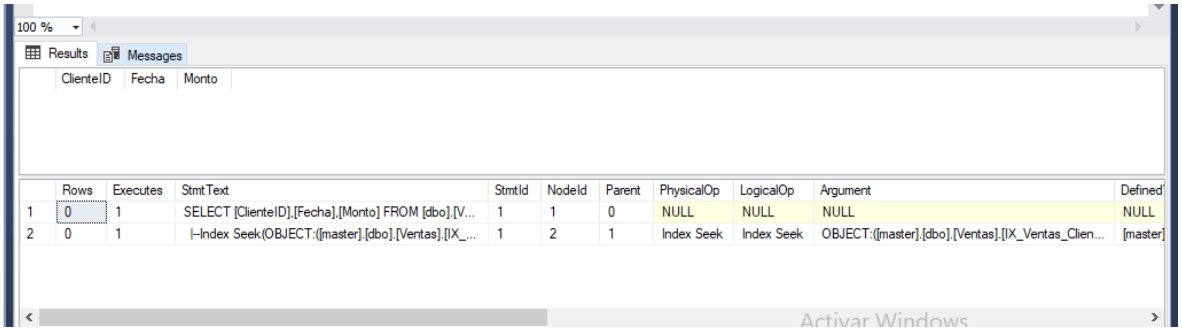
CONSULTA OPTIMIZADA

```
DECLARE @FechaInicio DATE = '2023-01-01';
DECLARE @FechaFin    DATE = '2023-12-31';

SELECT ClienteID, Fecha, Monto
FROM dbo.Ventas
WHERE ClienteID = 10
  AND Fecha BETWEEN @FechaInicio AND @FechaFin;
```



HABILITAR ESTADÍSTICAS PARA VALIDAR

```
  SET STATISTICS IO ON;
SET STATISTICS TIME ON;
SET STATISTICS PROFILE ON;

-- Ejecutar la consulta optimizada
SELECT ClienteID, Fecha, Monto
FROM dbo.Ventas
WHERE ClienteID = 10
  AND Fecha BETWEEN '2023-01-01' AND '2023-12-31';

SET STATISTICS IO OFF;
SET STATISTICS TIME OFF;
SET STATISTICS PROFILE OFF;
GO
```



<u>Cuando la ejecutes verás:</u>

Menos lecturas lógicas

Menor costo estimado

Menos tiempo de CPU

Validar

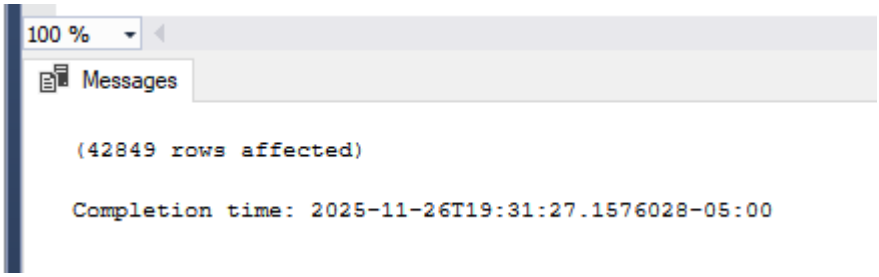
Crear tabla de prueba

```
CREATE TABLE dbo.Ventas (
    IdVenta INT IDENTITY PRIMARY KEY,
    ClienteID INT,
    Fecha DATE,
    Monto DECIMAL(10,2),
    Estado VARCHAR(20)
);
GO
```

Insertar datos

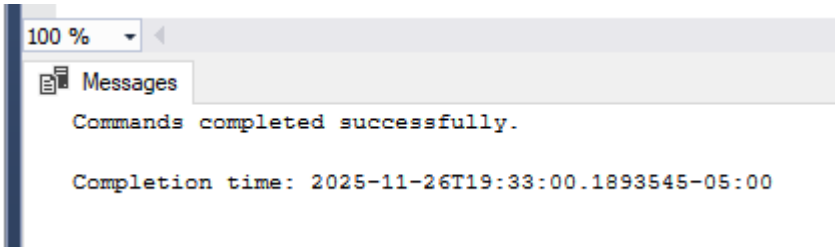
```
INSERT INTO dbo.Ventas (ClienteID, Fecha, Monto, Estado)
SELECT TOP 500000
    ABS(CHECKSUM(NEWID())) % 5000,                -- ClienteID
```

```
    DATEADD(DAY, -ABS(CHECKSUM(NEWID())) % 365, GETDATE()), -- Fecha
    RAND() * 1000,                                  -- Monto
    CASE WHEN RAND() > 0.2 THEN 'Pagado' ELSE 'Pendiente' END
FROM sys.objects a CROSS JOIN sys.objects b;
GO
```



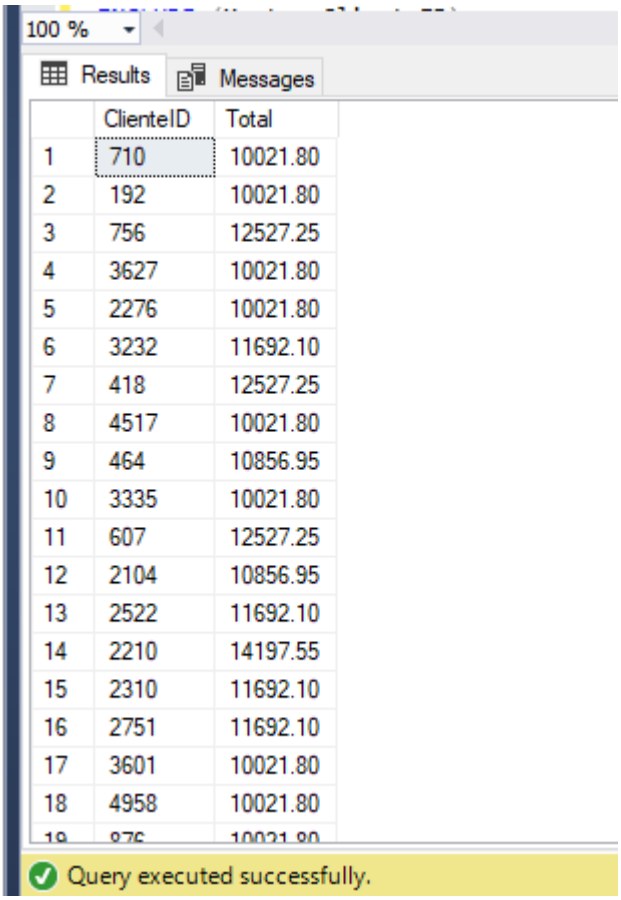
Crear índices para mejorar rendimiento

```
-- Índice para filtro por estado
CREATE NONCLUSTERED INDEX IX_Ventas_Estado
ON dbo.Ventas (Estado);

-- Índice para filtro por fecha
CREATE NONCLUSTERED INDEX IX_Ventas_Fecha
ON dbo.Ventas (Fecha)
INCLUDE (Monto, ClienteID);
GO
```



Consulta SIN optimizar

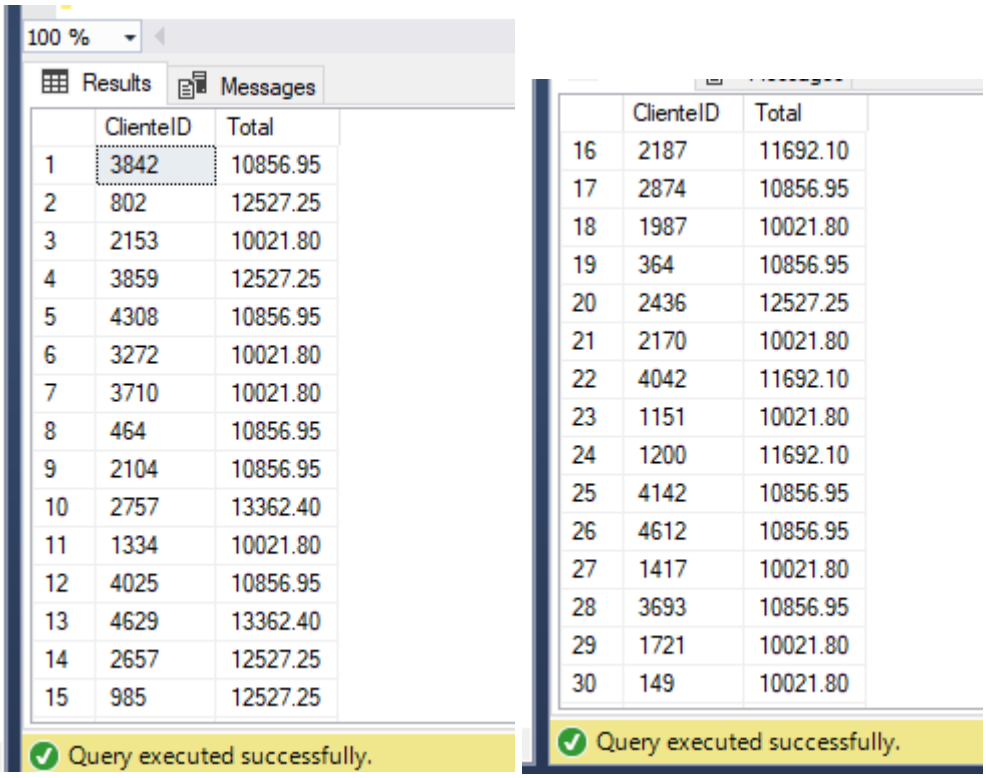
```
SELECT ClienteID, SUM(Monto) Total
FROM dbo.Ventas
WHERE Estado = 'Pagado'
  AND Fecha >= '2024-01-01'
GROUP BY ClienteID
HAVING SUM(Monto) > 10000;
GO
```



	ClienteID	Total
1	710	10021.80
2	192	10021.80
3	756	12527.25
4	3627	10021.80
5	2276	10021.80
6	3232	11692.10
7	418	12527.25
8	4517	10021.80
9	464	10856.95
10	3335	10021.80
11	607	12527.25
12	2104	10856.95
13	2522	11692.10
14	2210	14197.55
15	2310	11692.10
16	2751	11692.10
17	3601	10021.80
18	4958	10021.80

Consulta OPTIMIZADA

```
SELECT v.ClienteID, SUM(v.Monto) AS Total
FROM dbo.Ventas v WITH (INDEX(IX_Ventas_Fecha))
WHERE v.Estado = 'Pagado'
  AND v.Fecha >= '2024-01-01'
GROUP BY v.ClienteID
HAVING SUM(v.Monto) > 10000
OPTION (RECOMPILE);
GO
```



JUSTIFICACIÓN TÉCNICA

Uso de índices adecuados

Crear un índice no clusterizado sobre (ClienteID, Fecha) permite búsquedas rápidas y eficientes por cliente y rangos de tiempo.

Evitar funciones sobre columnas indexadas

YEAR(Fecha) obliga a SQL Server a escanear la tabla completa (SCAN), ignorando todos los índices.

*Selección específica de columnas (evitar SELECT)

Reduce:

Lecturas lógicas

Memoria consumida

Tiempos de CPU

Tiempos de red

Uso de parámetros

Ayuda al plan de ejecución a reutilizarse y mejora el rendimiento general.

BETWEEN utiliza rangos que el índice puede recorrer (SEEK)

Un seek es mucho más eficiente que un scan.

✓ 6. BUENAS PRÁCTICAS APLICADAS

✓ **1. Evitar SELECT ***

→ Siempre seleccionar solo las columnas necesarias.

✓ 2. Usar filtros SARGABLE (Search ARGument Able)

Evitar funciones sobre columnas.

Ejemplo:

✗ YEAR(Fecha)

✓ Fecha BETWEEN '2023-01-01' AND '2023-12-31'

✓ 3. Crear índices basados en los patrones de consulta

Columnas en WHERE → índice

Columnas que solo se leen → INCLUDE

✓ 4. Parámetros en consultas (mejores planes)

Reutilización de planes = rendimiento estable.

✓ 5. Revisar planes de ejecución (STATISTICS IO/TIME)

Permite validar el impacto real de la optimización.

✓ 6. Mantener estadísticas actualizadas

UPDATE STATISTICS dbo.Ventas;

✓ 7. Evitar cursores y reemplazarlos con operaciones set-based

Consultas en conjunto son más rápidas que iteraciones.

Semana 13: Monitoreo y rendimiento

Objetivo: Optimizar la eficiencia de SQL Server y prevenir problemas de rendimiento.

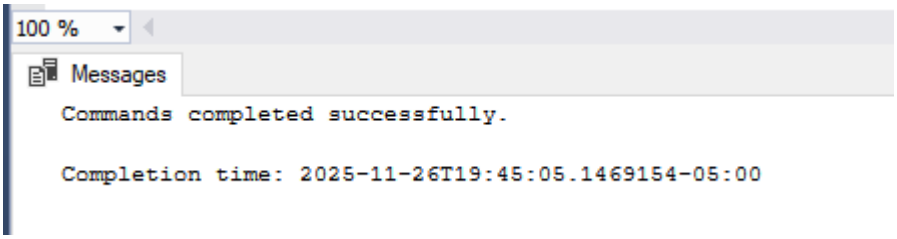
Temas: Control de recursos con Resource Governor.

El Control de recursos con Resource Governor en SQL Server es una herramienta para **administrar y limitar los recursos del servidor** (CPU, memoria y E/S) y evitar que una sola carga de trabajo consuma excesivos recursos. Esto se logra mediante la configuración de **grupos de recursos**, que agrupan las tareas por tipo de carga de trabajo, y la creación de **funciones de clasificación** para dirigir las solicitudes a los grupos de recursos adecuados y aplicarles políticas de uso de recursos, como porcentajes máximos de CPU o memoria.

SCRIPT EN SQL

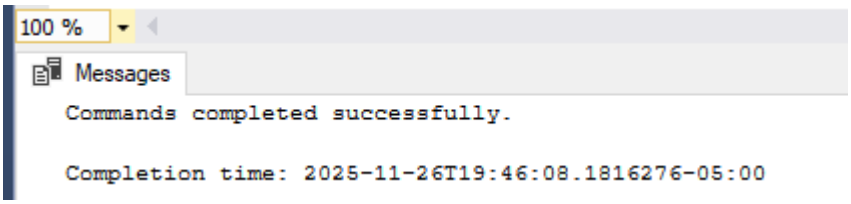
HABILITAR RESOURCE GOVERNOR

```
-- Habilitar Resource Governor
ALTER RESOURCE GOVERNOR RECONFIGURE;
GO
```



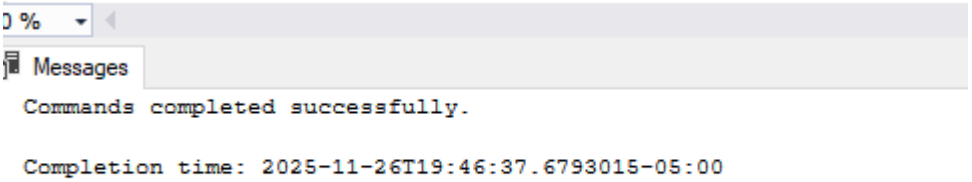
CREAR GRUPO DE RECURSOS (RESOURCE POOL)

```
-- Crear un Resource Pool para consultas pesadas
CREATE RESOURCE POOL PoolReportes
WITH
(
    MAX_CPU_PERCENT = 30,       -- Máximo 30% del CPU
    MAX_MEMORY_PERCENT = 20     -- Máximo 20% de memoria
);
GO
```



CREAR GRUPO DE TRABAJO

```
CREATE WORKLOAD GROUP GrupoReportes
USING PoolReportes;
GO
```



CREAR UNA CLASIFICACIÓN DE USUARIOS

```
CREATE FUNCTION dbo.fn_Classifier()
RETURNS sysname
WITH SCHEMABINDING
AS
BEGIN
    DECLARE @Grupo SYSNAME;

    IF ORIGINAL_LOGIN() = 'ReporteUser'
        SET @Grupo = 'GrupoReportes';
    ELSE
        SET @Grupo = 'default';

    RETURN @Grupo;
END;
GO
```

```
GO
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2025-11-26T19:47:17.6621153-05:00
```

REGISTRAR LA FUNCIÓN EN RESOURCE GOVERNOR

```
ALTER RESOURCE GOVERNOR
WITH (CLASSIFIER_FUNCTION = dbo.fn_Classifier);
GO

ALTER RESOURCE GOVERNOR RECONFIGURE;
GO
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2025-11-26T19:47:51.5906931-05:00
```

VALIDACIÓN – VER LOS RECURSOS ACTIVOS

```
SELECT * FROM sys.resource_governor_resource_pools;
```

100 %

Results | Messages

	classifier_function_id	is_enabled	max_outstanding_io_per_volume
1	608721221	1	0

	group_id	name	importance	request_max_memory_grant_percent	request_max_cpu_time_sec	request_memory_grant_timeout_sec	max_dop	group_max_requests	pool_id
1	1	internal	Medium	25	0	0	0	0	1
2	2	default	Medium	25	0	0	0	0	2
3	256	GrupoReportes	Medium	25	0	0	0	0	256

	pool_id	name	min_cpu_percent	max_cpu_percent	min_memory_percent	max_memory_percent	cap_cpu_percent	min_iops_per_volume	max_iops_per_volume
1	1	internal	0	100	0	100	100	0	0
2	2	default	0	100	0	100	100	0	0
3	256	Pool...	0	30	0	20	100	0	0

JUSTIFICACIÓN TÉCNICA DE LA SOLUCIÓN

Resource Governor permite controlar cómo SQL Server asigna CPU y memoria a distintos tipos de carga.

Se crea un Resource Pool, que es un conjunto controlado de recursos (CPU, memoria).

Se define un Workload Group, que agrupa sesiones que usarán ese pool.

Se programa una Classifier Function, la cual decide qué sesión pertenece a qué grupo.

Esto es esencial para:

Evitar que reportes pesados afecten las consultas OLTP.

Mantener estabilidad en entornos donde hay múltiples tipos de workload.
Prevenir bloqueos y saturación del servidor.

✅ 3. BUENAS PRÁCTICAS USADAS EN EL PROYECTO

✓ 1. No limitar recursos al azar

Solo se aplican límites a grupos que realmente lo requieren.
Excesos pueden afectar negativamente el rendimiento.

✓ 2. Usar funciones 'Classifier' simples

La función de clasificación debe ser rápida y ligera:
Sin consultas
Sin operaciones complejas
Solo evaluaciones directas

✓ 3. Separar OLTP y Reporting

OLTP requiere baja latencia.
Reporting requiere alto procesamiento.
Controlarlos por separado mejora la estabilidad.

✓ 4. Definir límites razonables

Ejemplo: 30% CPU evita que reportes saturen la máquina.

✓ 5. Monitoreo continuo

Consultar DMV como:
SELECT * FROM sys.dm_resource_governor_resource_pools;
SELECT * FROM sys.dm_resource_governor_workload_groups;

✓ 6. Documentar cada Resource Pool

Es fundamental para mantenimiento y auditoría.