

UNIVERSIDAD PERUANA LOS ANDES

**FACULTAD INGENIERIA INGENIERÍA DE
SISTEMAS Y COMPUTACIÓN**



**MANUAL DE LA PRACTICA DE
MONITOREO Y RENDIMIENTO EN SQL SERVER**

ASIGNATURA: BASE DE DATOS II

DOCENTE: MG. HERBERTH ARANDA ROJAS

ESTUDIANTE: Bonifacio Hilario Erick

CÓDIGO: S01238F

HUANCAYO-2025

Proyecto 1 — Captura de consultas lentas con Extended Events

1. Enunciado

Crear una sesión de Extended Events que capture consultas que tarden más de **1 segundo** en QhatuPeru, guardando en archivo .xel.

2. Script T-SQL

```
USE master;
GO

DECLARE @path NVARCHAR(400) = N'C:\XE\QhatuPeru_SlowQueries.xel'; --
ajusta ruta
DECLARE @dbname SYSNAME = N'QhatuPeru';
DECLARE @dbid INT = DB_ID(@dbname);
DECLARE @sessionname SYSNAME = N'XE_QhatuPeru_SlowQueries';
DECLARE @sql NVARCHAR(MAX);

-- Elimina sesión si existe
IF EXISTS (SELECT 1 FROM sys.server_event_sessions WHERE name =
@sessionname)
BEGIN
    ALTER EVENT SESSION [XE_QhatuPeru_SlowQueries] ON SERVER STATE =
STOP;
    DROP EVENT SESSION [XE_QhatuPeru_SlowQueries] ON SERVER;
END

-- Construir CREATE EVENT SESSION con valores embebidos
SET @sql = N'
CREATE EVENT SESSION ' + QUOTENAME(@sessionname) + N' ON SERVER
ADD EVENT sqlserver.sql_statement_completed(
    ACTION(sqlserver.sql_text, sqlserver.session_id,
sqlserver.database_id, sqlserver.client_app_name,
sqlserver.username)
    WHERE (sqlserver.database_id = ' + CAST(@dbid AS NVARCHAR(10)) +
N' AND duration > 1000000)
)
ADD TARGET package0.event_file(SET filename = N''' +
REPLACE(@path, '''', ''''''') + N'', max_file_size=(100),
max_rollover_files=(5))
WITH (MAX_MEMORY=4096 KB,
```

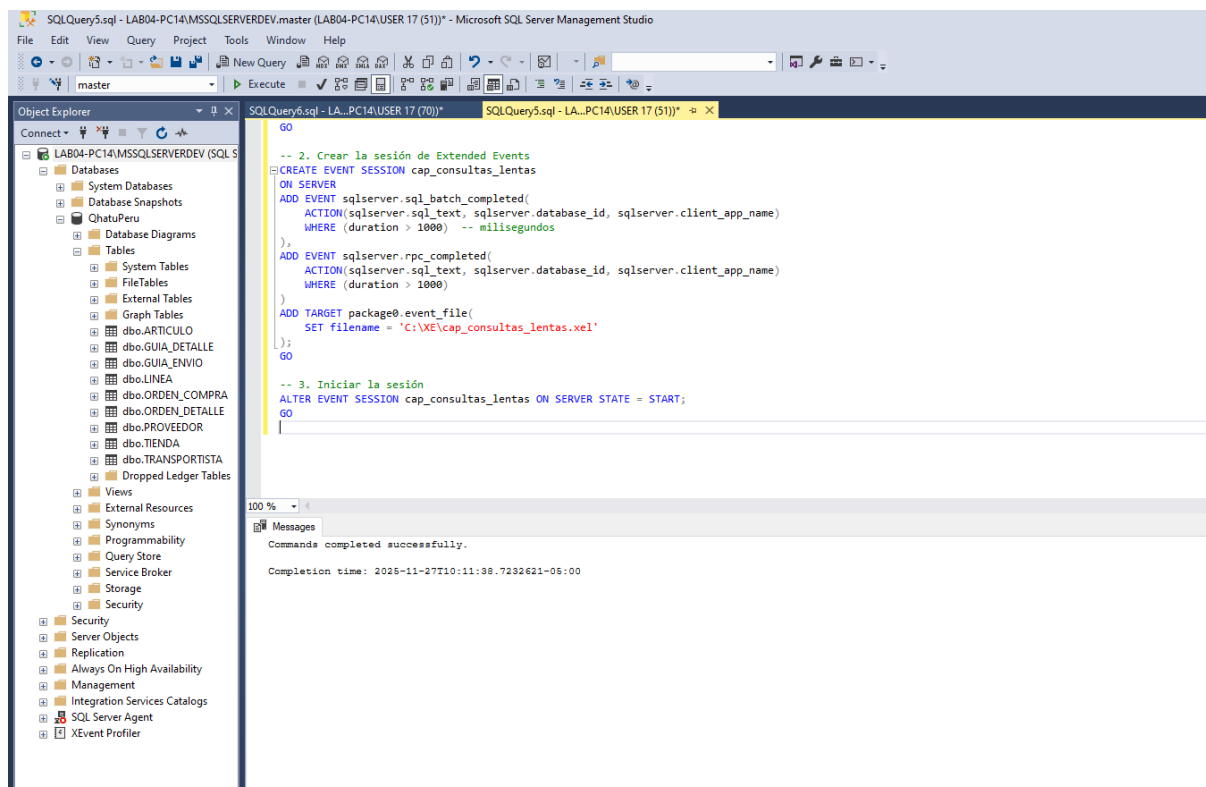
```
EVENT_RETENTION_MODE=ALLOW_SINGLE_EVENT_LOSS, MAX_DISPATCH_LATENCY=1
SECONDS);
';
```

```
EXEC sp_executesql @sql;
```

```
-- Iniciar sesión XE
```

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

```
GO
```



3. Justificación técnica

- duration está en microsegundos: $> 1,000,000 = 1s$.
- Uso SQL dinámico porque CREATE EVENT SESSION no acepta variables T-SQL directas para el filename ni para expresiones en WHERE.
- Se filtra por database_id para limitar captura.

| name | object_id | principal_id | schema_id | parent_object_id | type | type_desc | create_date | modify_date | is_ms_shipped | is_published | is_schema_published |
|----------------|-----------|--------------|-----------|------------------|------|--------------|-------------------------|-------------------------|---------------|--------------|---------------------|
| syscolumns | 3 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2022-10-08 06:28:41.360 | 2022-10-08 06:28:41.970 | 1 | 0 | 0 |
| sysobjects | 5 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2009-04-13 12:59:05.513 | 2022-10-08 06:28:41.607 | 1 | 0 | 0 |
| sysindexes | 6 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2022-10-08 06:28:41.550 | 2022-10-08 06:28:41.557 | 1 | 0 | 0 |
| sysallocunits | 7 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2009-04-13 12:59:05.500 | 2022-10-08 06:28:41.380 | 1 | 0 | 0 |
| sysfiles1 | 8 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2003-04-08 09:13:37.267 | 2003-04-08 09:13:37.267 | 1 | 0 | 0 |
| sysobjvalues | 9 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2022-10-08 06:28:41.637 | 2022-10-08 06:28:41.643 | 1 | 0 | 0 |
| sysindexeskeys | 16 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2022-10-08 06:28:41.560 | 2022-10-08 06:28:41.567 | 1 | 0 | 0 |
| sysproperties | 17 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2022-10-08 06:28:41.440 | 2022-10-08 06:28:41.453 | 1 | 0 | 0 |
| sysreflogs | 18 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2022-10-08 06:28:41.327 | 2022-10-08 06:28:41.337 | 1 | 0 | 0 |
| sysreflogs | 19 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2022-10-08 06:28:41.340 | 2022-10-08 06:28:41.347 | 1 | 0 | 0 |
| sysfiles | 20 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2022-10-08 06:28:41.520 | 2022-10-08 06:28:41.527 | 1 | 0 | 0 |
| syslogs | 21 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2022-10-08 06:28:41.330 | 2022-10-08 06:28:41.337 | 1 | 0 | 0 |
| syscheckfiles | 22 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2022-10-08 06:28:41.333 | 2022-10-08 06:28:41.340 | 1 | 0 | 0 |
| syslog | 23 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2022-10-08 06:28:41.340 | 2022-10-08 06:28:41.347 | 1 | 0 | 0 |
| sysfiles | 24 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2022-10-08 06:28:41.343 | 2022-10-08 06:28:41.350 | 1 | 0 | 0 |
| sysfiles | 25 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2009-04-13 12:59:05.810 | 2022-10-08 06:28:41.317 | 1 | 0 | 0 |
| sysnames | 27 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2009-04-13 12:59:05.797 | 2023-05-09 21:14:47.750 | 1 | 0 | 0 |
| sysnames | 28 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2009-04-13 12:59:05.593 | 2022-10-08 06:28:41.323 | 1 | 0 | 0 |
| sysnames | 29 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2009-04-13 12:59:05.450 | 2009-04-13 12:59:05.450 | 1 | 0 | 0 |
| sysnames | 34 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2009-04-13 12:59:05.687 | 2022-10-08 06:28:41.597 | 1 | 0 | 0 |
| sysnames | 35 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2022-10-08 06:28:41.673 | 2022-10-08 06:28:41.930 | 1 | 0 | 0 |
| sysnames | 36 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2022-10-08 06:28:41.710 | 2022-10-08 06:28:41.897 | 1 | 0 | 0 |
| sysnames | 37 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2022-10-08 06:28:41.717 | 2022-10-08 06:28:41.877 | 1 | 0 | 0 |
| sysnames | 38 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2022-10-08 06:28:41.717 | 2022-10-08 06:28:41.950 | 1 | 0 | 0 |
| sysnames | 39 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2009-04-13 12:59:07.967 | 2022-10-08 06:28:41.403 | 1 | 0 | 0 |
| sysnames | 40 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2022-10-08 06:28:41.740 | 2022-10-08 06:28:41.750 | 1 | 0 | 0 |
| sysnames | 41 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2009-04-13 12:59:05.733 | 2022-10-08 06:28:41.293 | 1 | 0 | 0 |
| sysnames | 42 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2009-04-13 12:59:05.640 | 2023-05-09 21:14:47.750 | 1 | 0 | 0 |
| sysnames | 43 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2009-04-13 12:59:05.610 | 2009-04-13 12:59:05.610 | 1 | 0 | 0 |
| sysnames | 44 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2009-04-13 12:59:05.717 | 2009-04-13 12:59:05.717 | 1 | 0 | 0 |
| sysnames | 45 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2009-04-13 12:59:05.653 | 2009-04-13 12:59:05.653 | 1 | 0 | 0 |
| sysnames | 46 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2009-04-13 12:59:08.217 | 2022-10-08 06:28:41.813 | 1 | 0 | 0 |
| sysnames | 47 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2009-04-13 12:59:05.623 | 2009-04-13 12:59:05.623 | 1 | 0 | 0 |
| sysnames | 48 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2009-04-13 12:59:05.640 | 2009-04-13 12:59:05.640 | 1 | 0 | 0 |
| sysnames | 49 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2009-04-13 12:59:05.827 | 2009-04-13 12:59:05.827 | 1 | 0 | 0 |
| sysnames | 50 | NULL | 4 | 0 | S | SYSTEM_TABLE | 2009-04-13 12:59:05.843 | 2009-04-13 12:59:05.843 | 1 | 0 | 0 |

4. Buenas prácticas

- Ajustar ruta a disco con espacio suficiente, usar rollover.
- Ejecutar con permisos adecuados y supervisar tamaño de archivos .xel.
- No capturar todo el sql_text si hay mucho volumen (podrías capturar solo query_hash en escenarios muy verbosos).

Proyecto 2 — Índices para búsqueda por DNI y Apellidos

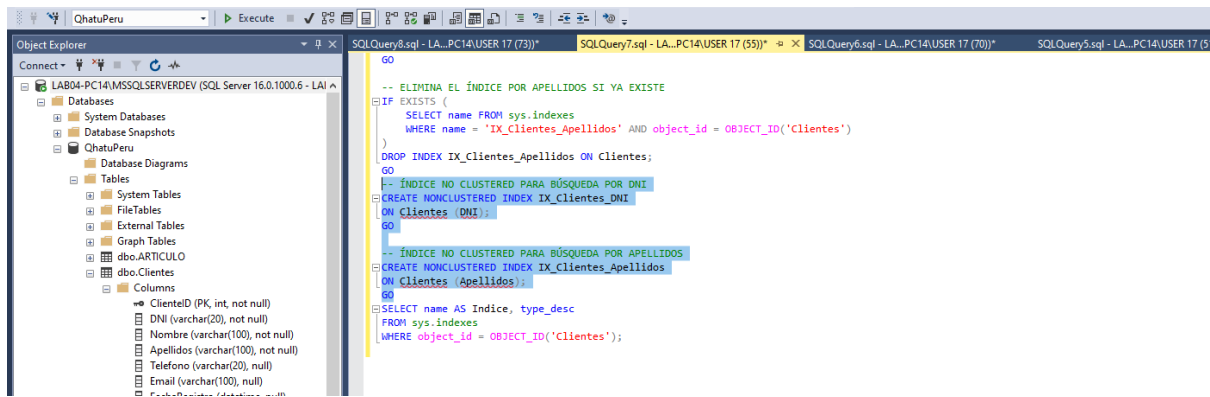
1. Enunciado

Crear índices para mejorar búsquedas por DNI y por Apellidos en dbo.Clientes.

2. Script T-SQL

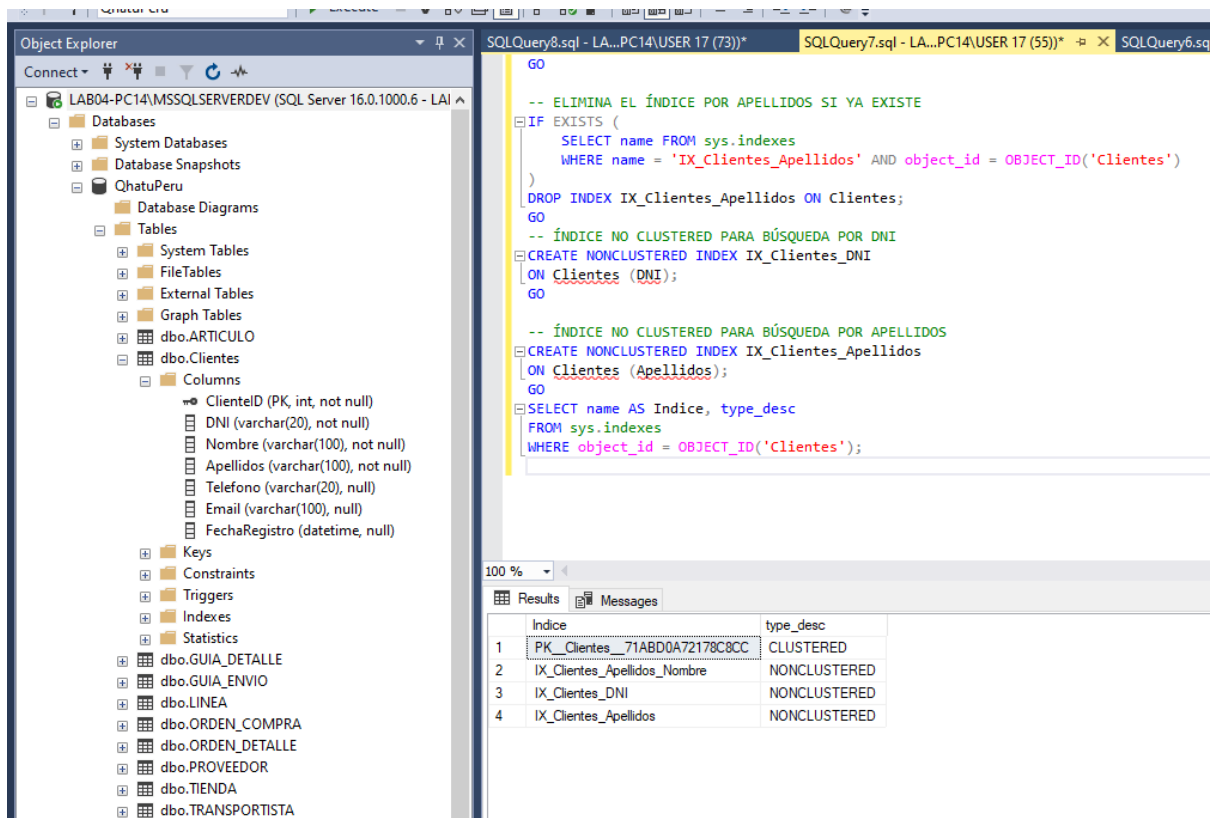
```
-- ÍNDICE NO CLUSTERED PARA BÚSQUEDA POR DNI CREATE NONCLUSTERED
INDEX IX_Clientes_DNI ON Clientes (DNI); GO
```

```
-- ÍNDICE NO CLUSTERED PARA BÚSQUEDA POR APELLIDOS CREATE
NONCLUSTERED INDEX IX_Clientes_Apellidos ON Clientes (Apellidos); GO
```



3. Justificación técnica

- DNI suele ser búsqueda exacta → índice único.
- (Apellidos, Nombre) mejora LIKE 'García%' y ORDER BY Apellidos, Nombre.
- INCLUDE evita lookups en consultas comunes.



4. Buenas prácticas

- Revisar cardinalidad antes de crear índices.
- Evitar índices redundantes; balance lectura/escritura.

Proyecto 3 — Detectar fragmentación y mantenimiento de índices

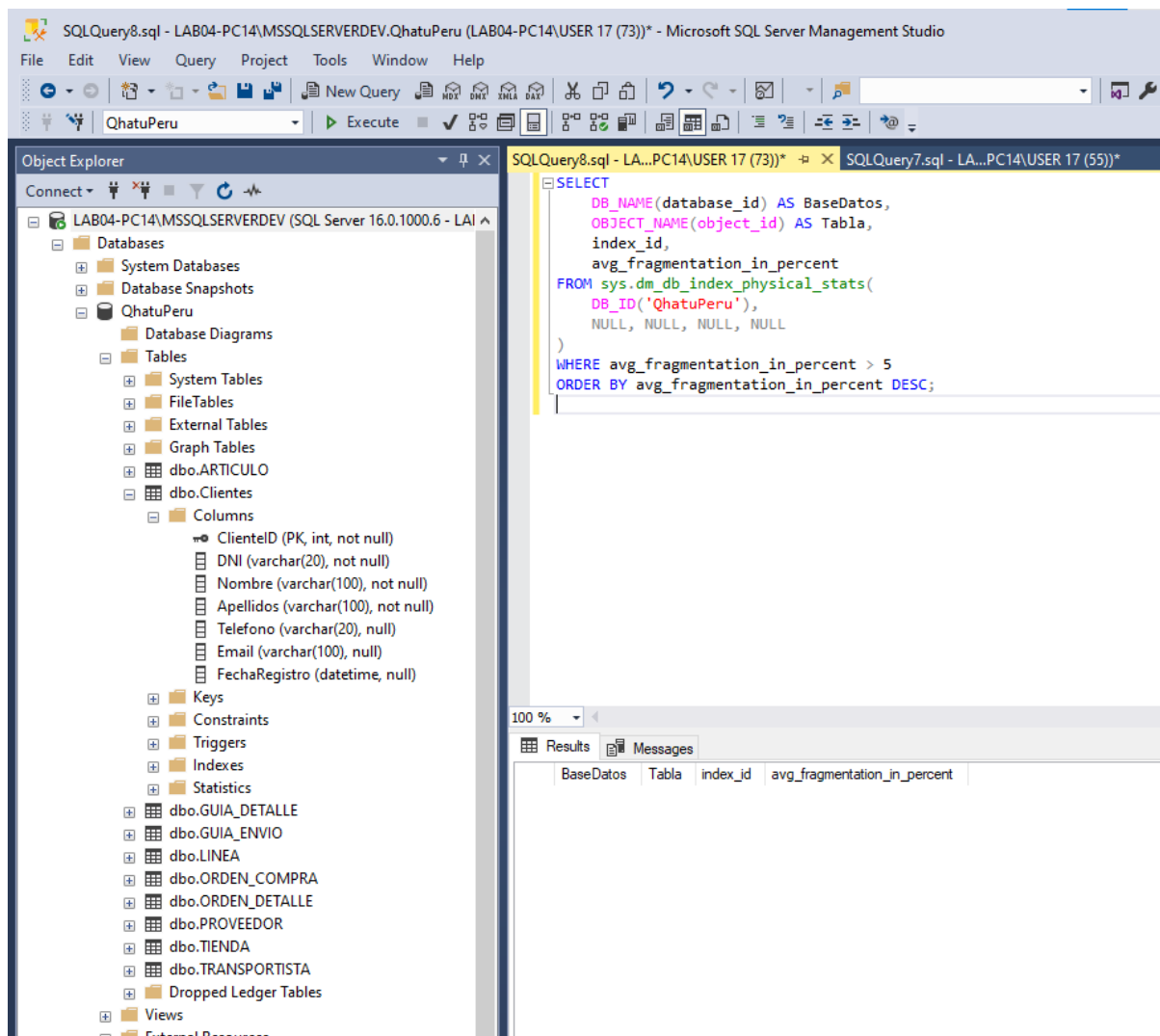
1. Enunciado

Evaluar fragmentación en QhatuPeru y REORGANIZE/REBUILD según porcentaje.

2. Script T-SQL

2.1 Detectar fragmentación

```
SELECT
    DB_NAME(database_id) AS BaseDatos,
    OBJECT_NAME(object_id) AS Tabla,
    index_id,
    avg_fragmentation_in_percent
FROM sys.dm_db_index_physical_stats(
    DB_ID('QhatuPeru'),
    NULL, NULL, NULL, NULL
)
WHERE avg_fragmentation_in_percent > 5
ORDER BY avg_fragmentation_in_percent DESC;
```



3. Justificación técnica

- avg_fragmentation_in_percent y page_count definen la acción.
- REORGANIZE menos costoso; REBUILD más efectivo.
- Construcción dinámica de ALTER INDEX evita errores de concatenación.

| database_id | object_id | index_id | partition_number | index_type_desc | alloc_unit_type_desc | index_depth | index_level | avg_fragmentation_in_percent | fragment_count | avg_fragment_size_in_pages | page_count | avg_page_space_used_in_percent | record_count | ghost |
|-------------|-----------|------------|------------------|--------------------|----------------------|-------------|-------------|------------------------------|----------------|----------------------------|------------|--------------------------------|--------------|-------|
| 1 | 5 | 901578250 | 1 | CLUSTERED INDEX | IN_ROW_DATA | 1 | 0 | 0 | 1 | 1 | 1 | 38.275356288009 | 50 | 0 |
| 2 | 5 | 933578364 | 1 | CLUSTERED INDEX | IN_ROW_DATA | 1 | 0 | 0 | 1 | 1 | 1 | 32.493204440007 | 50 | 0 |
| 3 | 5 | 933578364 | 2 | NONCLUSTERED INDEX | IN_ROW_DATA | 1 | 0 | 0 | 1 | 1 | 1 | 13.4544106745738 | 50 | 0 |
| 4 | 5 | 981578335 | 1 | CLUSTERED INDEX | IN_ROW_DATA | 1 | 0 | 0 | 1 | 1 | 1 | 65.962441314554 | 50 | 0 |
| 5 | 5 | 1013578649 | 1 | CLUSTERED INDEX | IN_ROW_DATA | 1 | 0 | 0 | 1 | 1 | 1 | 0.538967136150235 | 0 | 1 |
| 6 | 5 | 1108578991 | 1 | CLUSTERED INDEX | IN_ROW_DATA | 1 | 0 | 0 | 1 | 1 | 1 | 17.8897940980097 | 50 | 0 |
| 7 | 5 | 1141579105 | 1 | CLUSTERED INDEX | IN_ROW_DATA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 5 | 1205579333 | 1 | CLUSTERED INDEX | IN_ROW_DATA | 1 | 0 | 0 | 1 | 1 | 1 | 40.8450704225352 | 50 | 0 |
| 9 | 5 | 1237579447 | 1 | CLUSTERED INDEX | IN_ROW_DATA | 1 | 0 | 0 | 1 | 1 | 1 | 17.8897940980097 | 50 | 0 |
| 10 | 5 | 1381579679 | 1 | CLUSTERED INDEX | IN_ROW_DATA | 1 | 0 | 0 | 1 | 1 | 1 | 0.308870168479472 | 0 | 1 |
| 11 | 5 | 1385579903 | 1 | CLUSTERED INDEX | IN_ROW_DATA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 5 | 1385579903 | 2 | NONCLUSTERED INDEX | IN_ROW_DATA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 5 | 1385579903 | 3 | NONCLUSTERED INDEX | IN_ROW_DATA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 14 | 5 | 1385579903 | 4 | NONCLUSTERED INDEX | IN_ROW_DATA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 5 | 1977058079 | 1 | CLUSTERED INDEX | IN_ROW_DATA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16 | 5 | 1977058079 | 2 | NONCLUSTERED INDEX | IN_ROW_DATA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 5 | 2009059193 | 1 | CLUSTERED INDEX | IN_ROW_DATA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 5 | 2009059193 | 2 | NONCLUSTERED INDEX | IN_ROW_DATA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 5 | 2041058307 | 1 | CLUSTERED INDEX | IN_ROW_DATA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 5 | 2041058307 | 2 | NONCLUSTERED INDEX | IN_ROW_DATA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

4. Buenas prácticas

- Programar en ventanas de baja carga.
- Monitorear espacio tempdb/archivo de transacción (REBUILD puede ser intensivo).
- Log de operaciones.

Proyecto 4 — Transacción de venta (consistencia)

1. Enunciado

Simular transacción con INSERT en Ventas y UPDATE en Productos.Stock, garantizando consistencia.

2. Script T-SQL

BEGIN TRY

SET XACT_ABORT ON;

BEGIN TRAN trxVenta;

DECLARE @IdProducto INT = 101;

DECLARE @Cantidad INT = 2;

DECLARE @IdCliente INT = 2001;

DECLARE @Precio DECIMAL(18,2) = (SELECT PrecioVenta FROM
dbo.Productos WHERE ProductoId = @IdProducto);

IF @Precio IS NULL

THROW 50002, 'Producto no existe.', 1;

IF (SELECT Stock FROM dbo.Productos WHERE ProductoId =
@IdProducto) < @Cantidad


```

        THROW 50001, 'Stock insuficiente.', 1;

INSERT INTO dbo.Ventas (ClienteId, FechaVenta, Total)
VALUES (@IdCliente, SYSDATETIME(), @Precio * @Cantidad);

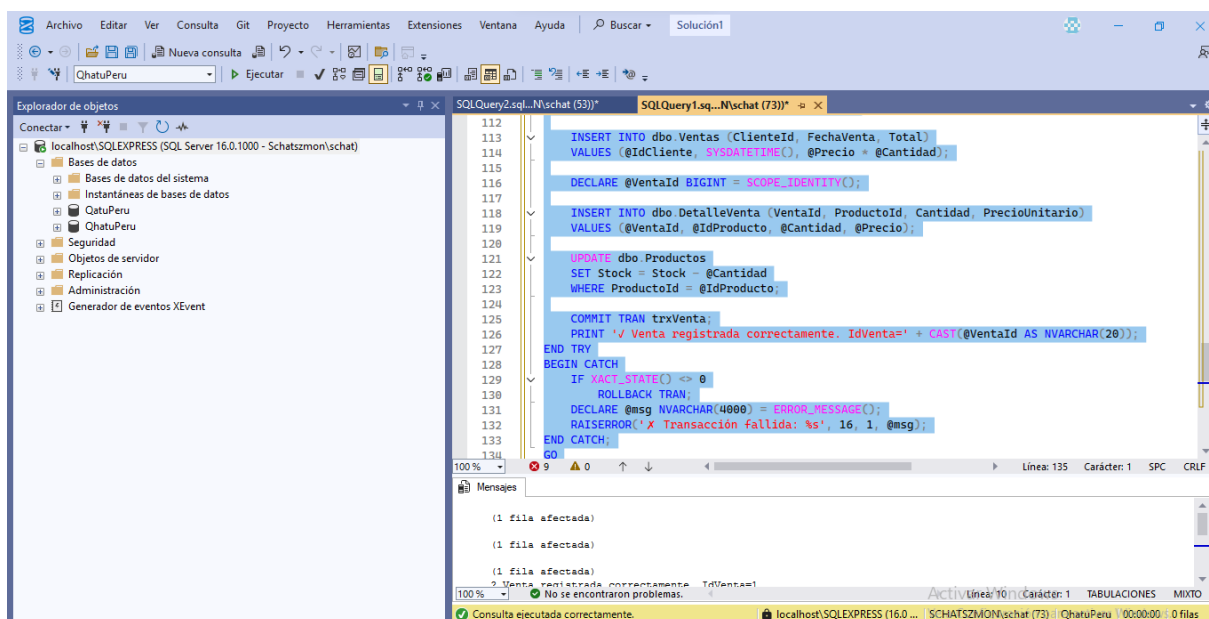
DECLARE @VentaId BIGINT = SCOPE_IDENTITY();

INSERT INTO dbo.DetalleVenta (VentaId, ProductoId, Cantidad,
PrecioUnitario)
VALUES (@VentaId, @IdProducto, @Cantidad, @Precio);

UPDATE dbo.Productos
SET Stock = Stock - @Cantidad
WHERE ProductoId = @IdProducto;

COMMIT TRAN trxVenta;
PRINT '✓ Venta registrada correctamente. IdVenta=' +
CAST(@VentaId AS NVARCHAR(20));
END TRY
BEGIN CATCH
    IF XACT_STATE() <> 0
        ROLLBACK TRAN;
    DECLARE @msg NVARCHAR(4000) = ERROR_MESSAGE();
    RAISERROR('X Transacción fallida: %s', 16, 1, @msg);
END CATCH;
GO

```



VERIFICAR RESULTADOS

```
PRINT '=== VENTAS REGISTRADAS ==='; SELECT * FROM dbo.Ventas;
```

```
PRINT '=== DETALLE DE VENTAS ==='; SELECT * FROM dbo.DetalleVenta;
```

```
PRINT '=== STOCK ACTUALIZADO ==='; SELECT ProductId, NombreProducto, Stock  
FROM dbo.Productos; GO
```

The screenshot shows a SQL Server query execution window. The query is as follows:

```
-- =====  
PRINT '=== VENTAS REGISTRADAS ===';  
SELECT * FROM dbo.Ventas;  
  
PRINT '=== DETALLE DE VENTAS ===';  
SELECT * FROM dbo.DetalleVenta;  
  
PRINT '=== STOCK ACTUALIZADO ===';  
SELECT ProductId, NombreProducto, Stock FROM dbo.Productos;  
GO
```

The results are displayed in three tables:

| | Ventaid | Clientid | FechaVenta | Total |
|---|---------|----------|-------------------------|--------|
| 1 | 1 | 2001 | 2025-12-03 07:53:12.960 | 170.00 |

| | DetalleId | Ventaid | ProductId | Cantidad | PrecioUnitario | Subtotal |
|---|-----------|---------|-----------|----------|----------------|----------|
| 1 | 1 | 1 | 101 | 2 | 85.00 | 170.00 |

| | ProductId | NombreProducto | Stock |
|---|-----------|------------------|-------|
| 1 | 100 | Laptop HP 15" | 10 |
| 2 | 101 | Mouse Logitech | 48 |
| 3 | 102 | Teclado Mecá... | 30 |
| 4 | 103 | Monitor Samsu... | 15 |
| 5 | 104 | Audífonos Sony | 25 |

3. Justificación técnica

- XACT_ABORT ON garantiza rollback ante errores severos.
- Validaciones previas evitan iniciar transacciones innecesarias.
- SCOPE_IDENTITY() para obtener id de la venta.

4. Buenas prácticas

- Mantener transacciones cortas.
- Control de concurrencia (optimistic/pessimistic según necesidad).
- Registrar auditoría si falla.

Proyecto 5 — Identificar bloqueos activos

1. Enunciado

Detectar sesiones que bloquean o están bloqueadas en el servidor.

2. Script T-SQL

```
USE master;

GO

-- Ver bloqueos activos

SELECT

    r.blocking_session_id AS BloqueoOrigen,

    r.session_id AS SesionBloqueada,

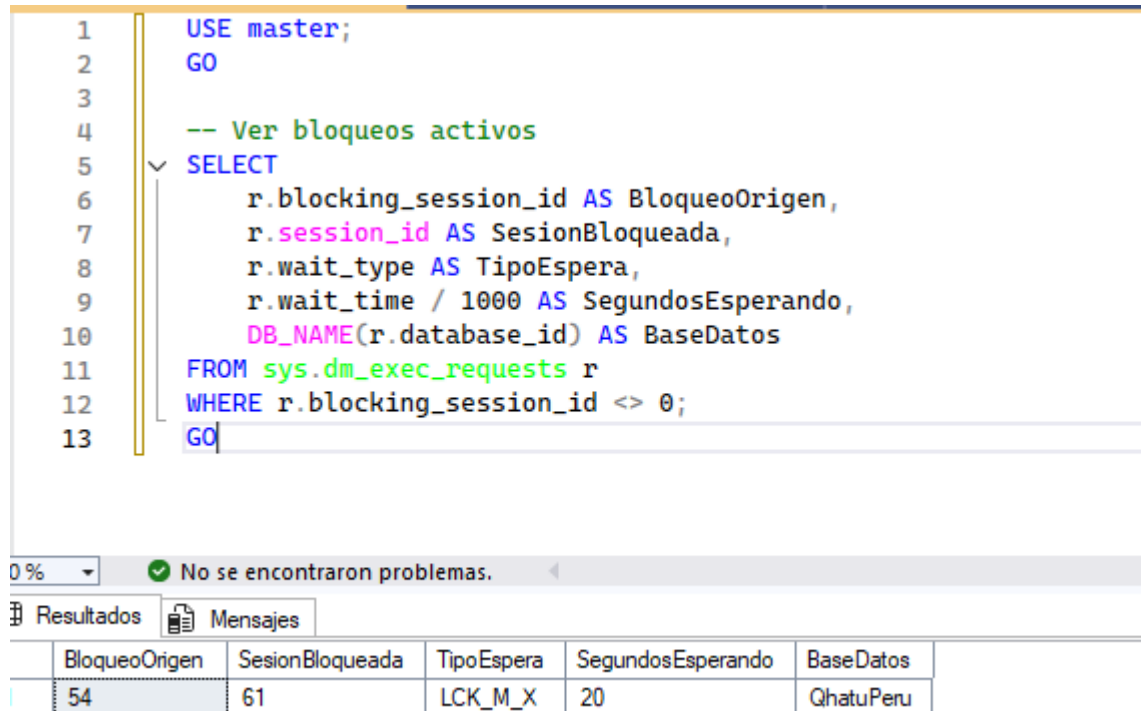
    r.wait_type AS TipoEspera,

    r.wait_time / 1000 AS SegundosEsperando,

    DB_NAME(r.database_id) AS BaseDatos

FROM sys.dm_exec_requests r

WHERE r.blocking_session_id <> 0.
```



The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays a T-SQL script with line numbers 1 through 13. The script is identical to the one in the previous block. The bottom pane shows the results of the query execution. A status bar at the top of the results pane indicates '0%' completion and 'No se encontraron problemas.' Below this, there are tabs for 'Resultados' and 'Mensajes'. The 'Resultados' tab is active, showing a table with 5 columns: BloqueoOrigen, SesionBloqueada, TipoEspera, SegundosEsperando, and BaseDatos. The table contains one row of data.

| BloqueoOrigen | SesionBloqueada | TipoEspera | SegundosEsperando | BaseDatos |
|---------------|-----------------|------------|-------------------|-----------|
| 54 | 61 | LCK_M_X | 20 | QhatuPeru |

3. Justificación técnica

- `sys.dm_exec_requests` y `sys.dm_os_waiting_tasks` permiten identificar SPIDs implicados y SQL ejecutado.
- Extrae SQL para investigar la causa.

4. Buenas prácticas

- No matar sesiones sin analizar; primero identificar transacción y usuario.

- Implementar alertas si bloqueo > umbral.

Proyecto 6 — Analizar plan de ejecución de consulta lenta

1. Enunciado

Analizar plan de ejecución de consulta que devuelve ventas por producto.

2. Script T-SQL

```
USE QhatuPeru;
```

```
GO
```

```
-- Obtener plan, IO y tiempo
```

```
SET STATISTICS XML ON;
```

```
SET STATISTICS IO ON;
```

```
SET STATISTICS TIME ON;
```

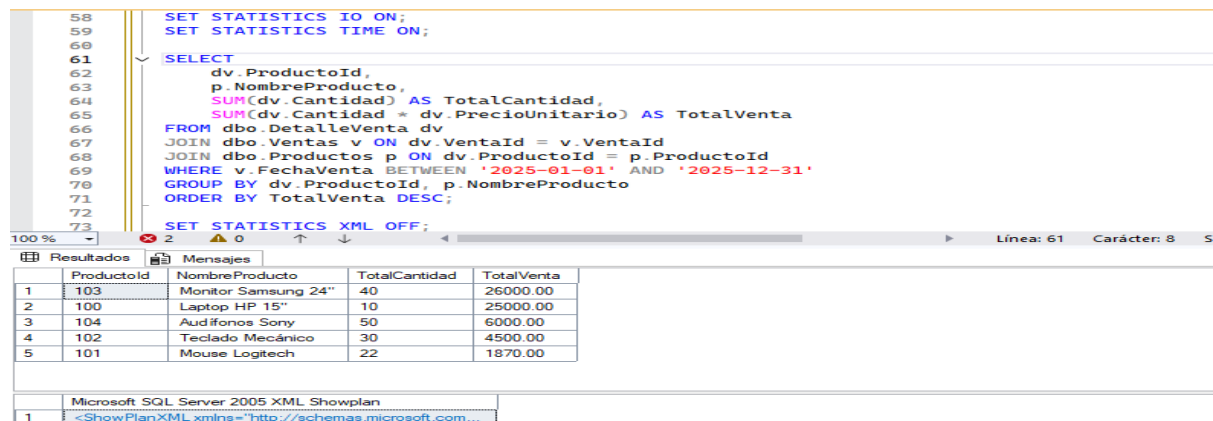
```
SELECT dv.ProductoId, SUM(dv.Cantidad) AS TotalCantidad,
SUM(dv.Cantidad * dv.PrecioUnitario) AS TotalVenta
FROM dbo.DetalleVenta dv
JOIN dbo.Ventas v ON dv.VentaId = v.VentaId
WHERE v.FechaVenta BETWEEN @f1 AND @f2 -- define variables o
reemplaza por fechas
GROUP BY dv.ProductoId
ORDER BY TotalVenta DESC;
```

```
SET STATISTICS XML OFF;
```

```
SET STATISTICS IO OFF;
```

```
SET STATISTICS TIME OFF;
```

```
GO
```



```

58 SET STATISTICS IO ON;
59 SET STATISTICS TIME ON;
60
61 SELECT
62     dv.ProductoId,
63     p.NombreProducto,
64     SUM(dv.Cantidad) AS TotalCantidad,
65     SUM(dv.Cantidad * dv.PrecioUnitario) AS TotalVenta
66 FROM   dbo.DetalleVenta dv
67 JOIN   dbo.Ventas v ON dv.VentaId = v.VentaId
68 JOIN   dbo.Productos p ON dv.ProductoId = p.ProductoId
69 WHERE  v.FechaVenta BETWEEN '2025-01-01' AND '2025-12-31'
70 GROUP BY dv.ProductoId
71 ORDER BY TotalVenta DESC;
72
73 SET STATISTICS XML OFF;

```

| | ProductoId | NombreProducto | TotalCantidad | TotalVenta |
|---|------------|---------------------|---------------|------------|
| 1 | 103 | Monitor Samsung 24" | 40 | 26000.00 |
| 2 | 100 | Laptop HP 15" | 10 | 25000.00 |
| 3 | 104 | Audífonos Sony | 50 | 6000.00 |
| 4 | 102 | Teclado Mecánico | 30 | 4500.00 |
| 5 | 101 | Mouse Logitech | 22 | 1870.00 |

Microsoft SQL Server 2005 XML Showplan

1 <ShowPlanXML xmlns="http://schemas.microsoft.com..."

3. Justificación técnica

- SET STATISTICS XML ON ofrece plan con estimaciones vs reales, spills, operaciones costosas.
- STATISTICS IO/TIME ayudan a cuantificar I/O y CPU.

4. Buenas prácticas

- Revisar mismatch estimado/real (posible falta de estadísticas).
- Probar índices/estadísticas en staging.

Proyecto 7 — Optimización con índices compuestos para filtrado por fecha y cliente

1. Enunciado

Optimizar consulta que filtra Ventas por FechaVenta y ClienteId.

2. Script T-SQL

```
USE QhatuPeru;  
GO
```

```
IF NOT EXISTS (SELECT 1 FROM sys.indexes WHERE name =  
'IX_Ventas_Cliente_Fecha' AND object_id = OBJECT_ID('dbo.Ventas'))  
BEGIN  
    CREATE NONCLUSTERED INDEX IX_Ventas_Cliente_Fecha  
    ON dbo.Ventas (ClienteId, FechaVenta)  
    INCLUDE (Total); -- cubrir SUM(Total)  
END  
GO
```

```
10 -- =====
11 -- 2. CREAR EL ÍNDICE OPTIMIZADO
12 -- =====
13 IF NOT EXISTS (SELECT 1 FROM sys.indexes WHERE name = 'IX_Ventas_Cliente_Fecha' AND object_id = 0)
14 BEGIN
15     CREATE NONCLUSTERED INDEX IX_Ventas_Cliente_Fecha
16     ON dbo.Ventas (ClienteId, FechaVenta)
17     INCLUDE (Total);
18
19     PRINT '✓ Índice IX_Ventas_Cliente_Fecha creado';
20 END
21 ELSE
22     PRINT 'El índice ya existe';
23 GO
24
25 -- =====
26 -- 3. VERIFICAR QUE SE CREÓ
```

Mensajes

? Índice IX_Ventas_Cliente_Fecha creado

Hora de finalización: 2025-12-03T08:21:19.7799711-05:00

```
27 -- =====
28 EXEC sp_helpindex 'dbo.Ventas';
29 GO
30
31 -- =====
32 -- 4. PROBAR CONSULTA OPTIMIZADA
33 -- =====
34 SET STATISTICS IO ON;
35 SET STATISTICS TIME ON;
```

| Resultados | | |
|----------------------------|---|-----------------------|
| index_name | index_description | index_keys |
| IX_Ventas_Cliente_Fecha | nonclustered located on PRIMARY | ClienteId, FechaVenta |
| PK_Ventas_5B4150AC027544C4 | clustered, unique, primary key located on PRIMARY | VentaId |

```
37 -- Consulta que usa el indice
38 SELECT
39     ClienteId,
40     COUNT(*) AS NumeroVentas,
41     SUM(Total) AS TotalVentas
42 FROM dbo.Ventas
43 WHERE ClienteId = 2001
44     AND FechaVenta BETWEEN '2025-01-01' AND '2025-12-31'
45 GROUP BY ClienteId;
46
47 SET STATISTICS IO OFF;
48 SET STATISTICS TIME OFF;
49 GO
50
51 -- =====
52 -- 5. VER SI USA EL ÍNDICE (plan de ejecución)
53 -- =====
54
```

100 % 1 0

Resultados Mensajes

| | ClienteId | NumeroVentas | TotalVentas |
|---|-----------|--------------|-------------|
| 1 | 2001 | 18 | 19900.00 |

3. Justificación técnica

- Orden (ClienteId, FechaVenta) favorece WHERE ClienteId = X AND FechaVenta BETWEEN
- INCLUDE (Total) evita lookups para la agregación.

4. Buenas prácticas

- Verificar patrón de consulta real; si la mayoría filtra por fecha primero, cambiar orden de columnas.

Proyecto 8 — Crear estadísticas manuales sobre Precio

1. Enunciado

Crear estadística manual sobre Precio en Productos para mejorar consultas rango.

2. Script T-SQL

```
USE QhatuPeru;
GO
```

```

-- Crear o actualizar STAT sobre Precio
IF NOT EXISTS (SELECT 1 FROM sys.stats WHERE object_id =
OBJECT_ID('dbo.Productos') AND name = 'STAT_Precio')
BEGIN
    CREATE STATISTICS STAT_Precio ON dbo.Productos (Precio) WITH
FULLSCAN;
END
ELSE
BEGIN
    UPDATE STATISTICS dbo.Productos (STAT_Precio) WITH FULLSCAN;
END
GO

```

```

10
11 -- 2. CREAR ESTADÍSTICA SOBRE PRECIOVENTA
12
13 IF NOT EXISTS (SELECT 1 FROM sys.stats WHERE object_id = OBJECT_ID('dbo.Productos') AND name = 'STAT_PrecioVenta')
14 BEGIN
15     CREATE STATISTICS STAT_PrecioVenta ON dbo.Productos (PrecioVenta) WITH FULLSCAN;
16     PRINT '✓ Estadística STAT_PrecioVenta creada';
17 END
18 ELSE
19 BEGIN
20     UPDATE STATISTICS dbo.Productos (STAT_PrecioVenta) WITH FULLSCAN;
21     PRINT '✓ Estadística STAT_PrecioVenta actualizada';
22 END
23 GO
24
25 -- 3. VERIFICAR QUE SE CREÓ
26
27
28 SELECT
29     s.name AS NombreEstadística,

```

100% 1 0 Línea: 13 Carácter: 1 SPC CRLF

Mensajes

? Estadística STAT_PrecioVenta creada

Hora de finalización: 2025-12-03T08:24:08.8757571-05:00

25
26 -- 3. VERIFICAR QUE SE CREÓ
27
28 SELECT
29 s.name AS NombreEstadística,
30 COL_NAME(s.object_id, sc.column_id) AS Columna,
31 s.auto_created AS AutoCreada,
32 s.user_created AS CreadaPorUsuario,
33 STATS_DATE(s.object_id, s.stats_id) AS UltimaActualizacion
34 FROM sys.stats s
35 JOIN sys.stats_columns sc ON s.stats_id = sc.stats_id AND s.object_id = sc.object_id
36 WHERE s.object_id = OBJECT_ID('dbo.Productos');
37 GO
38
39 -- 4. VER DETALLE DE LA ESTADÍSTICA
40
41

Resultados Mensajes

| NombreEstadística | Columna | AutoCreada | CreadaPorUsuario | UltimaActualizacion |
|------------------------------|-------------|------------|------------------|-------------------------|
| PK_Producto_A430AEA3043F79F0 | ProductId | 0 | 0 | 2025-12-03 07:53:12.903 |
| STAT_PrecioVenta | PrecioVenta | 0 | 1 | 2025-12-03 08:24:08.863 |

38
39 -- 4. VER DETALLE DE LA ESTADÍSTICA
40
41
42 DBCC SHOW_STATISTICS ('dbo.Productos', 'STAT_PrecioVenta');
43 GO
44
45
46 -- 5. PROBAR CONSULTA POR RANGO DE PRECIO
47
48 SET STATISTICS IO ON;
49
50 SELECT ProductId, NombreProducto, PrecioVenta
51 FROM dbo.Productos
52 WHERE PrecioVenta BETWEEN 100 AND 500;
53
54 SET STATISTICS IO OFF;
55 GO

100 % 1 0 Línea: 42 Carácter: 1 SPC CRLF

Resultados Mensajes

| | | | | | |
|---|---------|---|---|---|---|
| 3 | 650.00 | 0 | 1 | 0 | 1 |
| 4 | 2500.00 | 0 | 1 | 0 | 1 |

| | ProductId | NombreProducto | PrecioVenta |
|---|-----------|------------------|-------------|
| 1 | 102 | Teclado Mecánico | 150.00 |
| 2 | 104 | Audífonos Sony | 120.00 |

Activar Windows

3. Justificación técnica

- Estadísticas precisas permiten mejores estimaciones de cardinalidad para rangos (BETWEEN, >, <).

- FULLSCAN máxima precisión tras cargas masivas.

4. Buenas prácticas

- No ejecutar FULLSCAN demasiado seguido; hacerlo tras cargas importantes.
- Monitorear auto_update_statistics y evaluar si conviene mantener automático.

Proyecto 9 — Configuración de Resource Pool

(limitación CPU 20%)

1. Enunciado

Crear Resource Pool que limite el uso de CPU al 20% para consultas analíticas.

2. Script T-SQL

```
-- Requiere sysadmin
USE master;
GO

-- Crear pool y grupo
IF NOT EXISTS (SELECT 1 FROM sys.resource_governor_resource_pools
WHERE name = 'RG_Analitico_Pool')
BEGIN
    CREATE RESOURCE POOL RG_Analitico_Pool
    WITH (MIN_CPU_PERCENT = 0, MAX_CPU_PERCENT = 20, CAP_CPU_PERCENT
= 20);
END
GO

IF NOT EXISTS (SELECT 1 FROM sys.resource_governor_workload_groups
WHERE name = 'RG_Analitico_Group')
BEGIN
    CREATE WORKLOAD GROUP RG_Analitico_Group
    USING RG_Analitico_Pool;
END
GO

-- Función classifier simple
IF OBJECT_ID('dbo.RGClassifier', 'FN') IS NOT NULL
    DROP FUNCTION dbo.RGClassifier;
GO

CREATE FUNCTION dbo.RGClassifier()
RETURNS sysname
WITH SCHEMABINDING
AS
BEGIN
    DECLARE @grp sysname = 'default';
    IF APP_NAME() = 'AnalysisBatch' OR SUSER_SNAME() = 'analista'
```

```

        SET @grp = 'RG_Analitico_Group';
    RETURN @grp;
END;
GO

```

```

ALTER RESOURCE GOVERNOR WITH (CLASSIFIER_FUNCTION =
dbo.RGClassifier);
ALTER RESOURCE GOVERNOR RECONFIGURE;
GO

```

```

8  |  L  -----
9  |  IF NOT EXISTS (SELECT 1
10 |      FROM sys.resource_governor_resource_pools
11 |      WHERE name = 'RG_Analitico_Pool')
12 |  BEGIN
13 |      CREATE RESOURCE POOL RG_Analitico_Pool
14 |      WITH (
15 |          MIN_CPU_PERCENT = 0,
16 |          MAX_CPU_PERCENT = 20,
17 |          CAP_CPU_PERCENT = 20
18 |      );
19 |      PRINT '✓ Resource Pool RG_Analitico_Pool creado';
20 |  END
21 |  ELSE
22 |      PRINT 'Resource Pool ya existe';
23 |  GO
24 |
25 |  -----
26 |
27 |  L  -----
28 |
29 |  L  -----
30 |
31 |  L  -----
32 |
33 |  L  -----
34 |  Línea: 34  Carácter: 1
35 |
36 |  Mensajes
37 |
38 |  Resource Pool RG_Analitico_Pool creado
39 |
40 |  Hora de finalización: 2025-12-03T08:29:55.9043218-05:00

```

```

37 |  L  -----
38 |  IF NOT EXISTS (SELECT 1
39 |      FROM sys.resource_governor_workload_groups
40 |      WHERE name = 'RG_Analitico_Group')
41 |  BEGIN
42 |      CREATE WORKLOAD GROUP RG_Analitico_Group
43 |      USING RG_Analitico_Pool;
44 |      PRINT '✓ Workload Group RG_Analitico_Group creado';
45 |  END
46 |  ELSE
47 |      PRINT 'Workload Group ya existe';
48 |  GO
49 |
50 |  -----
51 |
52 |  L  -----
53 |
54 |  L  -----
55 |
56 |  L  -----
57 |
58 |  L  -----
59 |  Línea: 34  Carácter: 1
60 |
61 |  Mensajes
62 |
63 |  ? Workload Group RG_Analitico_Group creado
64 |
65 |  Hora de finalización: 2025-12-03T08:30:18.1373568-05:00

```

```
53 IF OBJECT_ID('dbo.RGClassifier', 'FN') IS NOT NULL
54     DROP FUNCTION dbo.RGClassifier;
55 GO
56
57 CREATE FUNCTION dbo.RGClassifier()
58 RETURNS sysname
59 WITH SCHEMABINDING
60 AS
61 BEGIN
62     DECLARE @grp sysname = 'default';
63
64     -- Si la app se llama 'AnalisisBatch' o el usuario es 'analista'
65     IF APP_NAME() = 'AnalisisBatch'
66        OR SUSER_SNAME() = 'analista'
67        SET @grp = 'RG_Analitico_Group';
68
69     RETURN @grp;
70 END;
71 GO
72
73 PRINT '✓ Función clasificadora creada';
74
```

00 % 1 0 Línea: 74 Carácter: 1

Mensajes

? Función clasificadora creada

Hora de finalización: 2025-12-03T08:30:39.4044160-05:00

```
89 SELECT
90     name AS NombrePool,
91     min_cpu_percent AS MinCPU,
92     max_cpu_percent AS MaxCPU,
93     cap_cpu_percent AS LimiteCPU
94 FROM sys.resource_governor_resource_pools;
95 GO
96
97 SELECT
98     name AS NombreGrupo,
99     pool_id AS PoolId
100 FROM sys.resource_governor_workload_groups;
101 GO
102
```

00 % 1 0

Resultados Mensajes

| | NombrePool | MinCPU | MaxCPU | LimiteCPU |
|---|-------------------|--------|--------|-----------|
| 1 | internal | 0 | 100 | 100 |
| 2 | default | 0 | 100 | 100 |
| 3 | RG_Analitico_Pool | 0 | 20 | 20 |

| | NombreGrupo | PoolId |
|---|----------------|--------|
| 1 | internal | 1 |
| 2 | default | 2 |
| 3 | RG_Analitic... | 256 |

3. Justificación técnica

- Resource Governor controla recursos por pool y grupo; CAP_CPU_PERCENT limita CPU.

- Classifier enruta sesiones analíticas.
- 4. Buenas prácticas**
- Probar en staging; función classifier debe ser ligera.
 - Monitorear impactos y ajustar porcentajes.

Proyecto 10 — Auditoría ligera de inserciones en Productos con Extended Events

1. Enunciado

Auditar inserciones en Productos usando Extended Events sin afectar rendimiento.

2. Script T-SQL (dinámico para ruta)

```
USE master;
GO
```

```
DECLARE @path NVARCHAR(400) = N'C:\XE\QhatuPeru_ProductInserts.xel';
-- ajusta
DECLARE @session SYSNAME = N'XE_QhatuPeru_ProductInserts';
DECLARE @dbname SYSNAME = N'QhatuPeru';
DECLARE @dbid INT = DB_ID(@dbname);
DECLARE @sql NVARCHAR(MAX);

-- Eliminar si existe
IF EXISTS (SELECT 1 FROM sys.server_event_sessions WHERE name =
@session)
BEGIN
    ALTER EVENT SESSION [XE_QhatuPeru_ProductInserts] ON SERVER
STATE = STOP;
    DROP EVENT SESSION [XE_QhatuPeru_ProductInserts] ON SERVER;
END

-- Crear sesión con filtro por database y por texto (mínimo)
SET @sql = N'
CREATE EVENT SESSION ' + QUOTENAME(@session) + N' ON SERVER
ADD EVENT sqlserver.sql_statement_completed(
    ACTION(sqlserver.sql_text, sqlserver.session_id,
sqlserver.database_id, sqlserver.username,
sqlserver.client_app_name)
    WHERE (sqlserver.database_id = ' + CAST(@dbid AS NVARCHAR(10)) +
```

```
N' AND sqlserver.sql_text LIKE ''%INSERT%INTO%Productos%'')
)
ADD TARGET package0.event_file(SET filename = N''' +
REPLACE(@path, '', '') + N'', max_file_size=(50),
max_rollover_files=(10))
WITH (MAX_MEMORY=2048 KB,
EVENT_RETENTION_MODE=ALLOW_SINGLE_EVENT_LOSS, MAX_DISPATCH_LATENCY=1
SECONDS);
';

EXEC sp_executesql @sql;

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

```

11 USE master;
12 GO
13
14 DECLARE @path NVARCHAR(400) = N'C:\XE\QhatuPeru_ProductInserts.xel';
15 DECLARE @session SYSNAME = N'XE_QhatuPeru_ProductInserts';
16 DECLARE @dbname SYSNAME = N'QhatuPeru';
17 DECLARE @dbid INT = DB_ID(@dbname);
18 DECLARE @sql NVARCHAR(MAX);
19
20 -- Eliminar si existe
21 IF EXISTS (SELECT 1 FROM sys.server_event_sessions WHERE name = @session)
22 BEGIN
23     ALTER EVENT SESSION [XE_QhatuPeru_ProductInserts] ON SERVER STATE = STOP;
24     DROP EVENT SESSION [XE_QhatuPeru_ProductInserts] ON SERVER;
25     PRINT '✓ Sesión anterior eliminada';
26 END
27
28 -- Crear sesión
29 SET @sql = N'
30 CREATE EVENT SESSION ' + QUOTENAME(@session) + ' ON SERVER
31 ADD EVENT sqlserver.sql_statement_completed(
32     ACTION(sqlserver.sql_text, sqlserver.session_id, sqlserver.database_id, sqlserver.username, s

```

Mensajes

? Sesión Extended Events creada
? Sesión iniciada

Hora de finalización: 2025-12-03T08:35:18.4746422-05:00

```

78 SELECT
79     event_data.value('(event/@timestamp)[1]', 'DATETIME2') AS FechaEvento,
80     event_data.value('(event/action[@name="username"]/value)[1]', 'NVARCHAR(100)') AS Usuario,
81     event_data.value('(event/action[@name="sql_text"]/value)[1]', 'NVARCHAR(MAX)') AS Consulta
82 FROM (
83     SELECT CAST(event_data AS XML) AS event_data
84     FROM sys.fn_xe_file_target_read_file('C:\XE\QhatuPeru_ProductInserts*.xel', NULL, NULL, NULL,
85 ) AS datos;
86 GO

```

Resultados

| | FechaEvento | Usuario | Consulta |
|---|-----------------------------|------------------|---|
| 1 | 2025-12-03 13:35:54.2920000 | Schatszmon\schat | INSERT INTO dbo.Productos (NombreProducto, Des... |
| 2 | 2025-12-03 13:35:54.2930000 | Schatszmon\schat | INSERT INTO dbo.Productos (NombreProducto, Des... |

3. Justificación técnica

- XE es menos intrusivo que triggers T-SQL pesados.
- Se filtra por database_id y por patrón de texto para reducir ruido.

4. Buenas prácticas

- Si necesitas capturar valores antes/después considera CDC o trigger mínimo que escriba a una cola asíncrona.
- Revisar frecuencia de archivos .xel y archivarlos.