# UNIVERSIDAD PERUANA LOS ANDES

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN


## PRACTICA 12

**ASIGNATURA**: Base de Datos II

**ESTUDIANTE**: Espejo Quispe Luis Enrique

**DOCENTE:** Mg. Raúl Fernandez Bejarano

**CICLO:** V

**SECCIÓN:** A1

## HYO-2025

**Proyecto 1: Captura de consultas lentas con Extended Events**

1. Enunciado del ejercicio

   Crear una sesión de Extended Events que capture consultas que tarden más de 1 segundo en ejecutarse en la base de datos QhatuPeru. Guardar el resultado en un archivo .xel.

---

**SCRIPT T-SQL**

```
SELECT DB_ID('QhatuPeru') AS DatabaseID


CREATE EVENT SESSION CapturaConsultasLentas_01
ON SERVER
ADD EVENT sqlserver.sql_statement_completed
(
    ACTION (sqlserver.sql_text, sqlserver.database_id)
    WHERE (duration > 1000 AND sqlserver.database_id = 5)
)
ADD TARGET package0.asynchronous_file_target
(
    SET filename = 'C:\XE\ConsultasLentas.xel'
)
WITH (STARTUP_STATE = ON);
```

```
.00 %
Messages
Commands completed successfully.

Completion time: 2025-11-27T10:16:15.7297994-05:00
```

## Justificación técnica

- **Extended Events** es más eficiente que SQL Trace para monitoreo.
- Capturar consultas lentas ayuda a identificar problemas de rendimiento.
- Se usa `duration > 1000` para filtrar consultas que tardan más de 1 segundo.

## Buenas prácticas

- Almacenar el archivo `.xel` en una unidad rápida.
- Limitar la captura a la base específica para evitar sobrecarga.
- Revisar periódicamente y detener la sesión si no se necesita.

**Proyecto 2: Crear índices para mejorar la búsqueda de clientes**

1. Enunciado del ejercicio

   En la tabla Clientes, mejorar el rendimiento de búsqueda por DNI y Apellidos creando índices adecuados.

---

**SCRIPT T-SQL**

```
--ENUNCIADO 02
CREATE NONCLUSTERED INDEX IX_Clientes_DNI ON Clientes(DNI);
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2025-11-27T10:19:11.4885452-05:00
```

## Justificación técnica

- Los índices no agrupados aceleran búsquedas por columnas específicas.
- `DNI` y `Apellidos` son columnas usadas en filtros frecuentes.

## Buenas prácticas

- Evitar crear índices redundantes.
- Monitorear el impacto en operaciones `INSERT` y `UPDATE`.
- Usar nombres descriptivos para los índices.

**Proyecto 3 — Detectar fragmentación y aplicar mantenimiento de índices**

1. Enunciado del ejercicio

   Usar DMV para evaluar la fragmentación de índices en QhatuPeru y reconstruir o reorganizar según el porcentaje encontrado.

**SCRIPT T-SQL**

```sql
--ENUNCIADO 03
CREATE NONCLUSTERED INDEX IX_Articulo_PrecioProveedor ON ARTICULO(PrecioProveedor);

INSERT INTO LINEA (NomLinea, Descripcion)
SELECT 'Linea ' + CAST(ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) AS VARCHAR(10)), 'Descripcion'
FROM sys.objects;

INSERT INTO PROVEEDOR (NomProveedor, Representante, Direccion, Ciudad, Departamento, CodigoPostal, Telefono, Fax)
SELECT 'Proveedor ' + CAST(ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) AS VARCHAR(10)), 'Rep', 'Dir', 'Ciudad', 'Depto', '00000', '999999999', '999999999'
FROM sys.objects;

INSERT INTO ARTICULO (CodLinea, CodProveedor, DescripcionArticulo, Presentacion, PrecioProveedor, StockActual, StockMinimo)
SELECT TOP 500
    ABS(CHECKSUM(NEWID())) % 50 + 1,  -- CodLinea aleatorio
    ABS(CHECKSUM(NEWID())) % 50 + 1,  -- CodProveedor aleatorio
    'Articulo ' + CAST(ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) AS VARCHAR(10)),
    'Caja',
    CAST(RAND(CHECKSUM(NEWID())) * 100 AS MONEY),
    CAST(RAND(CHECKSUM(NEWID())) * 50 AS SMALLINT),
    5

SELECT
    OBJECT_NAME(ips.object_id) AS Tabla,
    i.name AS Indice,
    ips.avg_fragmentation_in_percent AS Fragmentacion
FROM sys.dm_db_index_physical_stats(DB_ID(), NULL, NULL, NULL, 'SAMPLED') ips
JOIN sys.indexes i ON ips.object_id = i.object_id AND ips.index_id = i.index_id
WHERE OBJECT_NAME(ips.object_id) IN ('ARTICULO');

-- Si fragmentación entre 10% y 30% → REORGANIZE
ALTER INDEX IX_Articulo_PrecioProveedor ON ARTICULO REORGANIZE;

-- Si fragmentación > 30% → REBUILD
ALTER INDEX IX_Articulo_PrecioProveedor ON ARTICULO REBUILD WITH (ONLINE = ON);
```

83 %

Results | Messages

| | Tabla | Indice | Fragmentacion |
|---|---|---|---|
| 1 | ARTICULO | PK__ARTICULO__BEC6083775659D6A | 0 |
| 2 | ARTICULO | IX_Articulo_PrecioProveedor | 0 |

**Justificación técnica**

- La DMV `sys.dm_db_index_physical_stats` mide fragmentación.
- `REORGANIZE` es menos costoso y se usa para fragmentación moderada.
- `REBUILD` elimina fragmentación alta y actualiza estadísticas.

**Buenas prácticas**

- Ejecutar mantenimiento en horarios de baja carga.
- Usar `ONLINE = ON` para minimizar impacto (si la edición lo soporta).
- Actualizar estadísticas después de reconstrucción.

**Proyecto 4: Manejo de transacciones para prevenir inconsistencias**

1. Enunciado del ejercicio

   Simular una transacción de venta con dos operaciones: insertar en Ventas y actualizar Stock. La transacción debe garantizar consistencia.

**SCRIPT T-SQL**

```sql
--ENUNCIADO 04
BEGIN TRANSACTION;

BEGIN TRY
    -- Insertar una orden de compra
    INSERT INTO ORDEN_COMPRA (NumOrden, FechaOrden, FechaIngreso)
    VALUES (1001, GETDATE(), NULL);

    -- Insertar detalle de la orden
    INSERT INTO ORDEN_DETALLE (NumOrden, CodArticulo, PrecioCompra, CantidadSolicitada)
    VALUES (1001, 1, 50.00, 5);

    -- Actualizar stock del artículo
    UPDATE ARTICULO
    SET StockActual = StockActual - 5
    WHERE CodArticulo = 1;

    COMMIT TRANSACTION;
END TRY
BEGIN CATCH
    ROLLBACK TRANSACTION;
    PRINT 'Error en la transacción';
END CATCH;
```

91 %

Messages

```
(1 row affected)

(1 row affected)

(1 row affected)

Completion time: 2025-11-27T10:31:21.3203200-05:00
```

**Justificación técnica**

- `BEGIN TRANSACTION` asegura atomicidad.
- `TRY...CATCH` maneja errores y evita inconsistencias.

**Buenas prácticas**

- Validar stock antes de descontar.
- Registrar errores en tabla de auditoría.

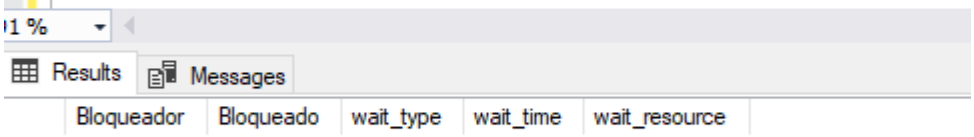
**Proyecto 5: Identificar bloqueos activos en la base QhatuPeru (PITR)**

1. Enunciado del ejercicio

   Detectar las sesiones que están bloqueando o siendo bloqueadas en el servidor.

**SCRIPT T-SQL**

```sql
SELECT
    blocking_session_id AS Bloqueador,
    session_id AS Bloqueado,
    wait_type,
    wait_time,
    wait_resource
FROM sys.dm_exec_requests
WHERE blocking_session_id <> 0;
```




## Justificación técnica

- DMV `sys.dm_exec_requests` muestra sesiones bloqueadas.
- Permite identificar problemas de concurrencia.

## Buenas prácticas

- Resolver bloqueos revisando transacciones largas.
- Evitar `LOCK HINTS` innecesarios.

**Proyecto 6: Analizar el plan de ejecución de una consulta lenta**

1. Enunciado del ejercicio

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

---

**SCRIPT T-SQL**

Cuando SHOWPLAN_XML ESTA ACTIVO:

```sql
--ENUNCIADO 06
SET SHOWPLAN_XML ON;

SELECT ProductoID, SUM(Cantidad) AS TotalVentas
FROM Ventas
GROUP BY ProductoID;

SET SHOWPLAN_XML OFF;
```

91 %

Results | Messages

| | Microsoft SQL Server 2005 XML Showplan |
|---|---|
| 1 | <ShowPlanXML xmlns="http://schemas.microsoft.com... |

CUANDO SHOWPLAN_XML ESTA DESACTIVADO

```sql
--ENUNCIADO 06
SET SHOWPLAN_XML ON;

SELECT CodArticulo, SUM(CantidadSolicitada) AS TotalSolicitado
FROM ORDEN_DETALLE
GROUP BY CodArticulo;

SET SHOWPLAN_XML OFF;
```

91 %

Results | Messages

| | CodArticulo | TotalSolicitado |
|---|---|---|
| 1 | 1 | 5 |

## Justificación técnica

- SHOWPLAN_XML permite ver el plan sin ejecutar la consulta.
- Identifica operaciones costosas (scans vs seeks).

## Buenas prácticas

- Usar índices adecuados para evitar `Table Scan`.
- Revisar estadísticas actualizadas.

**Proyecto 7: Optimización de consulta agregada con índices compuestos**

1. Enunciado del ejercicio

   Optimizar una consulta que filtra ventas por fecha y cliente.

**SCRIPT T-SQL**

```
--ENUNCIADO 07

CREATE NONCLUSTERED INDEX IX_OrdenDetalle_NumOrden_CodArticulo
ON ORDEN_DETALLE(NumOrden, CodArticulo);
```

```
Messages
Commands completed successfully.

Completion time: 2025-11-27T10:38:56.5276888-05:00
```

VERIFICACIÓN DEL INDICE

```
SELECT
    i.name AS NombreIndice,
    i.type_desc AS TipoIndice,
    c.name AS Columna,
    ic.key_ordinal AS OrdenEnIndice
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('ORDEN_DETALLE');
```

Results | Messages

| | NombreIndice | TipoIndice | Columna | OrdenEnIndice |
|---|---|---|---|---|
| 1 | PK_ORDEN_DETALLE | CLUSTERED | NumOrden | 1 |
| 2 | PK_ORDEN_DETALLE | CLUSTERED | CodArticulo | 2 |
| 3 | IX_OrdenDetalle_NumOrden_CodArticulo | NONCLUSTERED | NumOrden | 1 |
| 4 | IX_OrdenDetalle_NumOrden_CodArticulo | NONCLUSTERED | CodArticulo | 2 |

Justificación técnica

- Índice compuesto mejora consultas con filtros en ambas columnas.
- Reduce costo de lectura.

Buenas prácticas

- Ordenar columnas según selectividad.
- Evitar índices demasiado anchos.

**Proyecto 8: Creación de estadísticas manuales para mejorar el optimizador**

1. Enunciado del ejercicio

   Crear una estadística manual sobre la columna Precio en la tabla Productos para mejorar consultas de rango.

---

**SCRIPT T-SQL**

```
--ENUNCIADO 08
-- Crear estadística manual
CREATE STATISTICS Estadistica_PrecioProveedor ON ARTICULO(PrecioProveedor);


SELECT
    s.name AS NombreEstadistica,
    c.name AS Columna,
    s.auto_created,
    s.user_created,
    s.no_recompute
FROM sys.stats s
JOIN sys.stats_columns sc ON s.object_id = sc.object_id AND s.stats_id = sc.stats_id
JOIN sys.columns c ON sc.object_id = c.object_id AND sc.column_id = c.column_id
WHERE s.object_id = OBJECT_ID('ARTICULO');
```

91 %

Results | Messages

| | NombreEstadistica | Columna | auto_created | user_created | no_recompute |
|---|---|---|---|---|---|
| 1 | PK__ARTICULO__BEC6083775659D6A | CodArticulo | 0 | 0 | 0 |
| 2 | IX_Articulo_PrecioProveedor | PrecioProveedor | 0 | 0 | 0 |
| 3 | Estadistica_PrecioProveedor | PrecioProveedor | 0 | 1 | 0 |

## Justificación técnica

- Las estadísticas permiten al optimizador estimar la distribución de valores en una columna.
- Si el auto-creado no cubre todas las columnas, se pueden crear manualmente para mejorar el plan de ejecución.

## Buenas prácticas

- Actualizar estadísticas regularmente:
- Evitar exceso de estadísticas innecesarias.
- Monitorear impacto en consultas.

**Proyecto 9: Configuración de un Resource Pool para limitar recursos**

1. Enunciado del ejercicio

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

**SCRIPT T-SQL**

```
--ENUNCIADO 09

-- Crear Resource Pool
CREATE RESOURCE POOL PoolAnalitico
WITH (MAX_CPU_PERCENT = 20);

-- Crear Workload Group asociado
CREATE WORKLOAD GROUP GrupoAnalitico
USING PoolAnalitico;

-- Aplicar configuración
ALTER RESOURCE GOVERNOR RECONFIGURE;
```

91 %

Messages

```
Commands completed successfully.

Completion time: 2025-11-27T10:46:53.8766477-05:00
```

## Justificación técnica

- Resource Governor permite controlar el consumo de CPU, memoria y E/S por grupos de trabajo.
- Útil para evitar que consultas analíticas afecten procesos OLTP.

## Buenas prácticas

- Probar configuración en ambiente de desarrollo antes de producción.
- Monitorear impacto en rendimiento con DMV:

**Proyecto 10: Crear un trigger de auditoría ligera usando Extended Events**

1. Enunciado del ejercicio

   Auditar inserciones en la tabla Productos usando Extended Events sin afectar rendimiento.

**SCRIPT T-SQL**

```sql
--ENUNCIADO 10

-- Eliminar sesión si existe
IF EXISTS (SELECT * FROM sys.server_event_sessions WHERE name = 'AuditoriaInsertArticulo')
    DROP EVENT SESSION AuditoriaInsertArticulo ON SERVER;

-- Crear sesión de auditoría
CREATE EVENT SESSION AuditoriaInsertArticulo
ON SERVER
ADD EVENT sqlserver.sql_statement_starting
(
    ACTION (sqlserver.sql_text)
    WHERE sql_text LIKE '%INSERT INTO ARTICULO%'
)
ADD TARGET package0.asynchronous_file_target
(
    SET filename = 'C:\XE\AuditoriaArticulo.xel'
)
WITH (STARTUP_STATE = ON);

-- Iniciar sesión
ALTER EVENT SESSION AuditoriaInsertArticulo ON SERVER STATE = START;
```

```
%
Messages
Commands completed successfully.

Completion time: 2025-11-27T10:48:56.2016338-05:00
```

## Justificación técnica

- Extended Events es más eficiente que triggers tradicionales.
- Captura solo eventos relevantes sin bloquear operaciones.

## Buenas prácticas

- Limitar filtros para evitar sobrecarga.
- Revisar archivos periódicamente