“Año De La Recuperación Y
Consolidación De La Economía Peruana”

# UNIVERSIDAD PERUANA LOS ANDES

“FACULTAD DE INGENIERÍA”

ESCUELA PROFESIONAL “SISTEMAS Y COMPUTACIÓN”

## PRÁCTICA SEMANA 10

**CÁTEDRA:** Base de Datos II

**CATEDRÁTICO:** Ing. Fernandez Bejarano Raul Enrique

**ESTUDIANTE:** Quispe Segama Franklin Noe

**CICLO:** V

**SECCIÓN:** A1

**HUANCAYO PERÚ**

**2025**

# Proyecto 1: Creación y distribución de archivos físicos de la base QhatuPeru

### 1. Enunciado del ejercicio

Como parte de la expansión comercial de QhatuPeru, la empresa solicita la creación de una nueva base de datos que distribuya los datos entre un archivo primario, un archivo secundario y el archivo de registro de transacciones, asegurando mejor rendimiento y seguridad en el almacenamiento.

Ejercicio práctico:
a) Crear la base de datos QhatuPeru con un archivo primario, un archivo secundario y el log de transacciones en rutas distintas.
b) Consultar los archivos físicos asociados a la base QhatuPeru.

### 2. Script de la solución en T-SQL

```sql
--A) CREACIÓN DE LA BASE DE DATOS QhatuPERU
USE MASTER;
GO
IF EXISTS(SELECT *FROM sys.databases WHERE name ='QhatuPERU')
DROP DATABASE QhatuPERU;
GO

CREATE DATABASE QhatuPERU
ON PRIMARY
(
NAME = 'QhatuPERU_Primary',
FILENAME = 'C:\SQLData\QhatuPERU_Primary.mdf',
SIZE = 50MB,
MAXSIZE = 500MB,
FILEGROWTH = 10MB
),
FILEGROUP Secundario (
    NAME = QhatuPeru_Secundario,
    FILENAME = 'C:\SQLData\QhatuPERU_Secundario.ndf',
    SIZE = 100MB,
    MAXSIZE = 1GB,
    FILEGROWTH = 50MB
)
LOG ON (
    NAME = QhatuPeru_Log,
    FILENAME = 'C:\SQLLogs\QhatuPERU_Log.ldf',
    SIZE = 20MB,
    MAXSIZE = 500MB,
    FILEGROWTH = 10MB
);
GO
```

```sql
--B) Consulta de los archivos fisicos asociados a la base QhatuPERU
USE QhatuPERU;
GO
--1
EXEC sp_helpdb 'QhatuPERU';
--2
SELECT
    file_id,
    name AS NombreArchivo,
    physical_name AS RutaFisica,
    type_desc AS TipoArchivo,
    size * 8 / 1024 AS Tamaño_MB
FROM sys.database_files;
GO
--3
SELECT name, state_desc
FROM sys.databases
WHERE name = 'QhatuPERU';
```

100 %

Results | Messages

| | name | db_size | owner | dbid | created | status | compatibility_level |
|---|---|---|---|---|---|---|---|
| 1 | QhatuPERU | 170.00 MB | LAB04-PC09\USER 17 | 5 | Nov 6 2025 | Status=ONLINE, Updateability=READ_WRITE, UserAcc... | 160 |

| | name | fileid | filename | filegroup | size | maxsize | growth | usage |
|---|---|---|---|---|---|---|---|---|
| 1 | QhatuPERU_Primary | 1 | C:\SQLData\QhatuPERU_Primary.mdf | PRIMARY | 51200 KB | 512000 KB | 10240 KB | data only |
| 2 | QhatuPeru_Log | 2 | C:\SQLLogs\QhatuPERU_Log.ldf | NULL | 20480 KB | 512000 KB | 10240 KB | log only |
| 3 | QhatuPeru_Secundario | 3 | C:\SQLData\QhatuPERU_Secundario.ndf | Secundario | 102400 KB | 1048576 KB | 51200 KB | data only |

| | file_id | NombreArchivo | RutaFisica | TipoArchivo | Tamaño_MB |
|---|---|---|---|---|---|
| 1 | 1 | QhatuPERU_Primary | C:\SQLData\QhatuPERU_Primary.mdf | ROWS | 50 |
| 2 | 2 | QhatuPeru_Log | C:\SQLLogs\QhatuPERU_Log.ldf | LOG | 20 |
| 3 | 3 | QhatuPeru_Secundario | C:\SQLData\QhatuPERU_Secundario.ndf | ROWS | 100 |

| | name | state_desc |
|---|---|---|
| 1 | QhatuPERU | ONLINE |

### 3. Justificación técnica de la solución aplicada

- Se utilizó la instrucción **CREATE DATABASE** con la cláusula **ON** para definir los **archivos de datos (primario y secundario)** y la cláusula **LOG ON** para el archivo de **registro de transacciones**, permitiendo su almacenamiento en **rutas físicas distintas**, mejorando el rendimiento y la administración de almacenamiento.

- El **archivo primario (.mdf)** contiene los metadatos y objetos base del sistema; el **archivo secundario (.ndf)** amplía la capacidad de almacenamiento en otro disco físico; y el **archivo de log (.ldf)** guarda las transacciones, lo que mejora la recuperación ante fallos.

- El comando **sys.database_files** permite verificar la configuración física de los archivos asociados a la base de datos recién creada, garantizando la correcta distribución.

### 4. Explicación de las buenas prácticas utilizadas en el proyecto

- **Separación física de archivos:** colocar los archivos de datos y de log en discos distintos mejora el rendimiento de lectura/escritura y la recuperación ante fallos.
- **Asignación de tamaños iniciales razonables:** evita el crecimiento constante de archivos que puede degradar el rendimiento.
- **Definición de crecimiento controlado (FILEGROWTH):** se establecieron incrementos fijos (en MB) en lugar de porcentajes, lo que mantiene un crecimiento predecible.
- **Uso de nombres descriptivos:** facilita la administración y comprensión de los archivos en entornos con múltiples bases.
- **Validación posterior:** se usó una consulta a ***`sys.database_files`*** para confirmar que los archivos se crearon correctamente y están en las rutas esperadas.

# Proyecto 2: Ajuste de configuración y validación de propiedades de QhatuPeru

### 1. Enunciado del ejercicio

**Contexto:** QhatuPeru planea expandirse en el mercado nacional, por lo que solicita al equipo técnico optimizar las propiedades de la base de datos para mejorar compatibilidad y rendimiento. Se requiere cambiar la **colación** para soportar tildes y caracteres especiales, además de ajustar la **configuración de crecimiento automático** del archivo de datos para evitar saturaciones inesperadas.

**Ejercicio práctico:**
a) Consultar las propiedades actuales, modificar la colación y configurar el crecimiento automático del archivo principal.
b) Modificar el crecimiento automático del archivo primario de datos a **20 MB**.

### 2. Script de la solución en T-SQL

```sql
--Ejercicio 2
-- a) Consulta de las propiedades actuales de la base de datos QhatuPeru
USE QhatuPERU;
GO
SELECT
    name AS NombreArchivo,
    type_desc AS TipoArchivo,
    physical_name AS RutaFisica,
    size * 8 / 1024 AS Tamaño_MB,
    growth AS Crecimiento,
    is_percent_growth AS CrecimientoPorcentual
FROM sys.database_files;
GO
```

100 %

Results | Messages

| | NombreArchivo | TipoArchivo | RutaFisica | Tamaño_MB | Crecimiento | CrecimientoPorcentual |
|---|---|---|---|---|---|---|
| 1 | QhatuPERU_Primary | ROWS | C:\SQLData\QhatuPERU_Primary.mdf | 50 | 2560 | 0 |
| 2 | QhatuPeru_Log | LOG | C:\SQLLogs\QhatuPERU_Log.ldf | 20 | 1280 | 0 |
| 3 | QhatuPeru_Secundario | ROWS | C:\SQLData\QhatuPERU_Secundario.ndf | 100 | 6400 | 0 |

SQLQuery1.sql - LA...PC09\USER 17 (56))*

```sql
-- Modificación de la colación para soportar tildes y caracteres especiales
-- (Por ejemplo: Modern_Spanish_CI_AS admite acentos y ñ)
ALTER DATABASE QhatuPeru COLLATE Modern_Spanish_CI_AS;
GO

-- Configuración inicial del crecimiento automático del archivo principal
ALTER DATABASE QhatuPeru
MODIFY FILE (
    NAME = QhatuPeru_Primary,
    FILEGROWTH = 10MB
);
GO

-- b) Modificación del crecimiento automático del archivo primario de datos a 20 MB
ALTER DATABASE QhatuPeru
MODIFY FILE (
    NAME = QhatuPeru_Primary,
    FILEGROWTH = 20MB
);
GO

-- Verificación de los cambios aplicados
SELECT
    name AS NombreArchivo,
    physical_name AS RutaFisica,
    size * 8 / 1024 AS Tamaño_MB,
    growth AS ValorCrecimiento,
    CASE
        WHEN is_percent_growth = 1 THEN 'Porcentaje'
        ELSE 'Megabytes'
    END AS TipoCrecimiento,
    type_desc AS TipoArchivo
FROM sys.database_files;
GO
```

100 %

Results | Messages

| | NombreArchivo | TipoArchivo | RutaFisica | Tamaño_MB | Crecimiento | CrecimientoPorcentual |
|---|---|---|---|---|---|---|
| 1 | QhatuPERU_Primary | ROWS | C:\SQLData\QhatuPERU_Primary.mdf | 50 | 2560 | 0 |
| 2 | QhatuPeru_Log | LOG | C:\SQLLogs\QhatuPERU_Log.ldf | 20 | 1280 | 0 |
| 3 | QhatuPeru_Secundario | ROWS | C:\SQLData\QhatuPERU_Secundario.ndf | 100 | 6400 | 0 |

## 3. Justificación técnica de la solución aplicada

- Se consultaron las propiedades actuales mediante la vista del sistema **sys.database_files**, lo que permite conocer el tamaño, tipo de archivo y configuración de crecimiento.

- Se modificó la **colación** a `Modern_Spanish_CI_AS` porque esta admite caracteres especiales del idioma español (acentos, ñ y mayúsculas/minúsculas insensibles), garantizando integridad en la manipulación de textos.

- Se ajustó el parámetro **FILEGROWTH** para definir el crecimiento controlado del archivo de datos, primero a 10 MB y luego a 20 MB, previniendo saturaciones y evitando que el motor crezca el archivo en porcentajes impredecibles.

- Los comandos `ALTER DATABASE ... MODIFY FILE` permiten realizar estas modificaciones sin necesidad de recrear la base de datos, asegurando una configuración más flexible y administrable.

**4. Explicación de las buenas prácticas utilizadas en el proyecto**

- **Uso de colación acorde al idioma:** elegir `Modern_Spanish_CI_AS` asegura compatibilidad con tildes, eñes y ordenación alfabética correcta para entornos hispanos.
- **Crecimiento controlado en MB:** evita fragmentación y consumo excesivo de espacio en disco, brindando un rendimiento más estable.
- **Verificación posterior de configuración:** se ejecutó una consulta final a `sys.database_files` para confirmar que los cambios fueron aplicados correctamente.
- **Ejecución en entorno de mantenimiento:** las modificaciones de colación deben realizarse cuando la base no está en uso, para evitar bloqueos o inconsistencias.
- **Documentación de cada cambio:** se registran los scripts y resultados, permitiendo trazabilidad y soporte en auditorías futuras.

# Proyecto 3: Definición de modelo de recuperación y respaldo para QhatuPeru

**1. Enunciado del ejercicio**

En el área de operaciones de **QhatuPeru**, se identificaron diferentes necesidades de **recuperación de datos** según los procesos críticos de negocio. Por ello, se requiere configurar el **modelo de recuperación adecuado** y ejecutar un **respaldo completo** que garantice la seguridad e integridad de la información.

**Ejercicio práctico:**
a) Cambiar el modelo de recuperación de **QhatuPeru** a **SIMPLE** y luego a **BULK_LOGGED**, explicando la diferencia práctica entre ambos modelos.
b) Realizar un **respaldo completo** después de cambiar al modelo **FULL**.

## 2. Script de la solución en T-SQL

```sql
--Ejercicio 3
-- a) Consultar el modelo de recuperación actual
SELECT
    name AS NombreBaseDatos,
    recovery_model_desc AS ModeloRecuperacion
FROM sys.databases
WHERE name = 'QhatuPERU';
GO
```

100 %

Results | Messages

| | NombreBaseDatos | ModeloRecuperacion |
|---|---|---|
| 1 | QhatuPERU | FULL |

```sql
-- Cambiar el modelo de recuperación a SIMPLE
ALTER DATABASE QhatuPERU
SET RECOVERY SIMPLE;
GO

-- Verificar el cambio
SELECT
    name AS NombreBaseDatos,
    recovery_model_desc AS ModeloRecuperacion
FROM sys.databases
WHERE name = 'QhatuPERU';
GO
```

100 %

Results | Messages

| | NombreBaseDatos | ModeloRecuperacion |
|---|---|---|
| 1 | QhatuPERU | SIMPLE |

```sql
-- Cambiar el modelo de recuperación a BULK_LOGGED
ALTER DATABASE QhatuPERU
SET RECOVERY BULK_LOGGED;
GO

-- Verificar nuevamente
SELECT
    name AS NombreBaseDatos,
    recovery_model_desc AS ModeloRecuperacion
FROM sys.databases
WHERE name = 'QhatuPERU';
GO
```

0 %

Results | Messages

| | NombreBaseDatos | ModeloRecuperacion |
|---|---|---|
| 1 | QhatuPERU | BULK_LOGGED |

```sql
-- b) Cambiar el modelo a FULL para ejecutar el respaldo completo
ALTER DATABASE QhatuPERU
SET RECOVERY FULL;
GO

-- Crear respaldo completo de la base de datos
BACKUP DATABASE QhatuPERU
TO DISK = 'C:\SQLBackups\QhatuPERU_FullBackup.bak'
WITH
    FORMAT,
    NAME = 'RespaldoCompleto_QhatuPERU',
    SKIP,
    INIT,
    STATS = 10;
GO
```

0 %

Messages

```
23 percent processed.
46 percent processed.
69 percent processed.
92 percent processed.
100 percent processed.
Processed 552 pages for database 'QhatuPERU', file 'QhatuPERU_Primary' on file 1.
Processed 24 pages for database 'QhatuPERU', file 'QhatuPeru_Secundario' on file 1.
Processed 1 pages for database 'QhatuPERU', file 'QhatuPeru_Log' on file 1.
BACKUP DATABASE successfully processed 577 pages in 0.018 seconds (250.081 MB/sec).

Completion time: 2025-11-06T10:17:45.6721555-05:00
```

### 3. Justificación técnica de la solución aplicada

- Se utilizó la vista del sistema **sys.databases** para verificar el modelo de recuperación actual antes y después de cada cambio, asegurando que las modificaciones fueron efectivas.

- El modelo de recuperación **SIMPLE** se aplica cuando no es necesario mantener un registro detallado de las transacciones; el espacio del log se reutiliza automáticamente, reduciendo mantenimiento y consumo de disco.

- El modelo **BULK_LOGGED** es intermedio: permite operaciones masivas (como importaciones o indexaciones) con registro mínimo en el log, mejorando el rendimiento en cargas de datos grandes, aunque con menor granularidad en la restauración.

- Finalmente, se estableció el modelo **FULL** antes de realizar el respaldo completo, ya que este modelo conserva todo el historial de transacciones, permitiendo restauraciones punto en el tiempo en caso de fallos.

- El comando **BACKUP DATABASE** genera una copia completa del estado actual de la base de datos y se almacena en una ruta segura (`D:\SQLBackups`), garantizando la disponibilidad para recuperación.

**4. Explicación de las buenas prácticas utilizadas en el proyecto**

- **Verificación antes y después de los cambios:** se usaron consultas a `sys.databases` para asegurar que cada modelo de recuperación se aplicó correctamente.
- **Uso de rutas dedicadas para respaldos:** almacenar las copias en un disco distinto a los archivos de datos y log mejora la seguridad ante fallos físicos.
- **Cambio controlado de modelo de recuperación:** se planificaron los cambios según la necesidad operativa: *SIMPLE* para entornos de prueba, *BULK_LOGGED* para cargas masivas, y *FULL* para producción y respaldo completo.
- **Nombrado y formato del backup:** se incluyó `WITH FORMAT` y `NAME` para facilitar la trazabilidad del respaldo.
- **Documentación y registro:** mantener los scripts y resultados en bitácoras garantiza auditoría y soporte ante incidentes.

# Proyecto 4: Implementación de roles y usuarios para seguridad en QhatuPeru

**1. Enunciado del ejercicio**

Con el crecimiento de los equipos de **ventas** y **atención al cliente**, QhatuPeru requiere fortalecer la **seguridad** a nivel de base de datos. Se solicita la creación de **usuarios con roles diferenciados**: los **cajeros** solo podrán registrar ventas, los **usuarios de atención al cliente** podrán consultar información, y los **administradores o gerentes** tendrán control total o acceso a reportes.

**Ejercicio práctico:**
a) Crear el usuario `VendedorQhatu` y asignarlo al rol **db_datawriter** para registrar ventas.
b) Crear el usuario `ConsultaCliente` y asignarlo solo al rol **db_datareader**.

### 2. Script de la solución en T-SQL

```sql
--Ejercicio 4
USE QhatuPeru;
GO

-- a) Crear el inicio de sesión (Login) a nivel de servidor
CREATE LOGIN VendedorQhatu WITH PASSWORD = 'Vendedor@2025';
GO

-- Crear el usuario dentro de la base de datos y asignarlo al rol db_datawriter
CREATE USER VendedorQhatu FOR LOGIN VendedorQhatu;
GO
EXEC sp_addrolemember 'db_datawriter', 'VendedorQhatu';
GO

-- b) Crear el inicio de sesión (Login) para el usuario de consultas
CREATE LOGIN ConsultaCliente WITH PASSWORD = 'Cliente@2025';
GO

-- Crear el usuario dentro de la base de datos y asignarlo al rol db_datareader
CREATE USER ConsultaCliente FOR LOGIN ConsultaCliente;
GO
EXEC sp_addrolemember 'db_datareader', 'ConsultaCliente';
GO

-- (Opcional) Verificar roles asignados a cada usuario
SELECT
    dp.name AS Usuario,
    rp.name AS RolAsignado
FROM sys.database_role_members drm
JOIN sys.database_principals rp ON drm.role_principal_id = rp.principal_id
JOIN sys.database_principals dp ON drm.member_principal_id = dp.principal_id;
GO
```

100 %

Results | Messages

| | Usuario | RolAsignado |
|---|---|---|
| 1 | dbo | db_owner |
| 2 | ConsultaCliente | db_datareader |
| 3 | VendedorQhatu | db_datawriter |

**3. Justificación técnica de la solución aplicada**

- Se utilizó el comando **CREATE LOGIN** para generar los inicios de sesión a nivel de servidor SQL Server, lo que permite la autenticación de usuarios.

- Posteriormente, con **CREATE USER**, se asoció cada inicio de sesión con la base de datos **QhatuPeru**, permitiendo el acceso a los objetos internos.

- El usuario **VendedorQhatu** fue agregado al rol **db_datawriter**, que concede permisos de **INSERT**, **UPDATE** y **DELETE**, adecuados para registrar y actualizar ventas.

- El usuario **ConsultaCliente** fue asignado al rol **db_datareader**, otorgándole permisos **solo de lectura (SELECT)**, garantizando que no pueda modificar información.

- Finalmente, se verificó la asignación mediante una consulta a las vistas del sistema `sys.database_role_members` y `sys.database_principals`.

**4. Explicación de las buenas prácticas utilizadas en el proyecto**

- **Principio de privilegio mínimo:** cada usuario recibe solo los permisos necesarios para su función (lectura o escritura), reduciendo el riesgo de errores o accesos indebidos.
- **Separación de logins y usuarios:** se gestionan logins (nivel servidor) y usuarios (nivel base de datos) de forma independiente, manteniendo una arquitectura de seguridad más limpia.
- **Nombres descriptivos y contraseñas seguras:** se usaron nombres claros y contraseñas con mayúsculas, minúsculas y símbolos para cumplir políticas de seguridad.
- **Asignación mediante roles integrados:** en lugar de otorgar permisos directos, se usaron los roles predefinidos `db_datareader` y `db_datawriter`, simplificando la administración y auditoría.
- **Validación posterior:** se realizó una consulta para confirmar que las asignaciones de roles fueron aplicadas correctamente.

# Proyecto 5: Configuración granular de permisos en el módulo de ventas de QhatuPeru

**1. Enunciado del ejercicio**

El **gerente de ventas** necesita revisar información del módulo de ventas sin poder modificarla. El **equipo técnico** debe configurar permisos específicos que garanticen el **principio de mínimo privilegio**, permitiendo acceso solo a los datos estrictamente necesarios.

**Ejercicio práctico:**

a) Otorgar a `GerenteQhatu` acceso exclusivo (solo **SELECT**) a la tabla **Reportes**.

b) Revocar a `CajeroQhatu` el permiso **UPDATE** sobre la tabla **Ventas**.

**2. Script de la solución en T-SQL**

```sql
--Ejercicio 5
USE QhatuPeru;
GO

-- a) Crear el inicio de sesión y usuario para el Ge
IF NOT EXISTS (SELECT * FROM sys.server_principals WH
BEGIN
    CREATE LOGIN GerenteQhatu WITH PASSWORD = 'Geren
END;
GO

IF NOT EXISTS (SELECT * FROM sys.database_principals
BEGIN
    CREATE USER GerenteQhatu FOR LOGIN GerenteQhatu;
END;
GO

--b) Crear al inicio de sesion y usuario a Cajero (s

IF NOT EXISTS (SELECT * FROM sys.database_principals
BEGIN
    CREATE USER CajeroQhatu FOR LOGIN CajeroQhatu;
END;
GO

IF NOT EXISTS (SELECT * FROM sys.server_principals WH
BEGIN
    CREATE LOGIN CajeroQhatu WITH PASSWORD = 'Cajero(
END;
GO

-- Otorgar permiso de solo lectura (SELECT) sobre la
GRANT SELECT ON dbo.Reportes TO GerenteQhatu;
GO

-- b) Revocar permiso de actualización (UPDATE) a Ca
REVOKE UPDATE ON dbo.VENTAS FROM CajeroQhatu;
GO

-- (Opcional) Verificar permisos actuales de los usu
SELECT
    dp.name AS Usuario,
    o.name AS Objeto,
    p.permission_name AS Permiso,
    p.state_desc AS Estado
FROM sys.database_permissions p
JOIN sys.database_principals dp ON p.grantee_princip
JOIN sys.objects o ON p.major_id = o.object_id
WHERE dp.name IN ('GerenteQhatu', 'CajeroQhatu');
GO
```

Results | Messages

| | Usuario | Objeto | Permiso | Estado |
|---|---|---|---|---|
| 1 | GerenteQhatu | REPORTES | SELECT | GRANT |

### 3. Justificación técnica de la solución aplicada

- Se creó el inicio de sesión y usuario **GerenteQhatu** en la base de datos *QhatuPeru* para establecer la autenticación y acceso local a los objetos.

- Con el comando **GRANT SELECT**, se otorgó acceso de **solo lectura** a la tabla `Reportes`, garantizando que el gerente pueda **consultar** información sin posibilidad de modificarla, cumpliendo el principio de seguridad de solo lectura.

- Posteriormente, mediante **REVOKE UPDATE**, se eliminó el permiso de actualización del usuario **CajeroQhatu** sobre la tabla `Ventas`, evitando que realice cambios directos en los registros.

- La vista del sistema **sys.database_permissions** se utilizó para verificar qué permisos están activos en la base de datos y confirmar que las asignaciones fueron aplicadas correctamente.

**4. Explicación de las buenas prácticas utilizadas en el proyecto**

- **Principio de mínimo privilegio:** se asignaron permisos estrictamente necesarios, evitando otorgar privilegios innecesarios que podrían comprometer la integridad de los datos.
- **Permisos a nivel de objeto:** en lugar de roles globales, se aplicaron permisos específicos (SELECT y REVOKE UPDATE) a tablas concretas (`Reportes` y `Ventas`), logrando un control granular.
- **Creación condicional de usuarios:** se usaron condiciones `IF NOT EXISTS` para evitar errores en caso de que los usuarios ya existan.
- **Separación de funciones:** se diferencia claramente entre usuarios operativos (Cajero) y administrativos (Gerente), alineando la seguridad con la estructura organizacional.
- **Verificación de permisos:** la consulta final permite auditar los permisos concedidos o revocados, ayudando al control interno y cumplimiento de políticas.

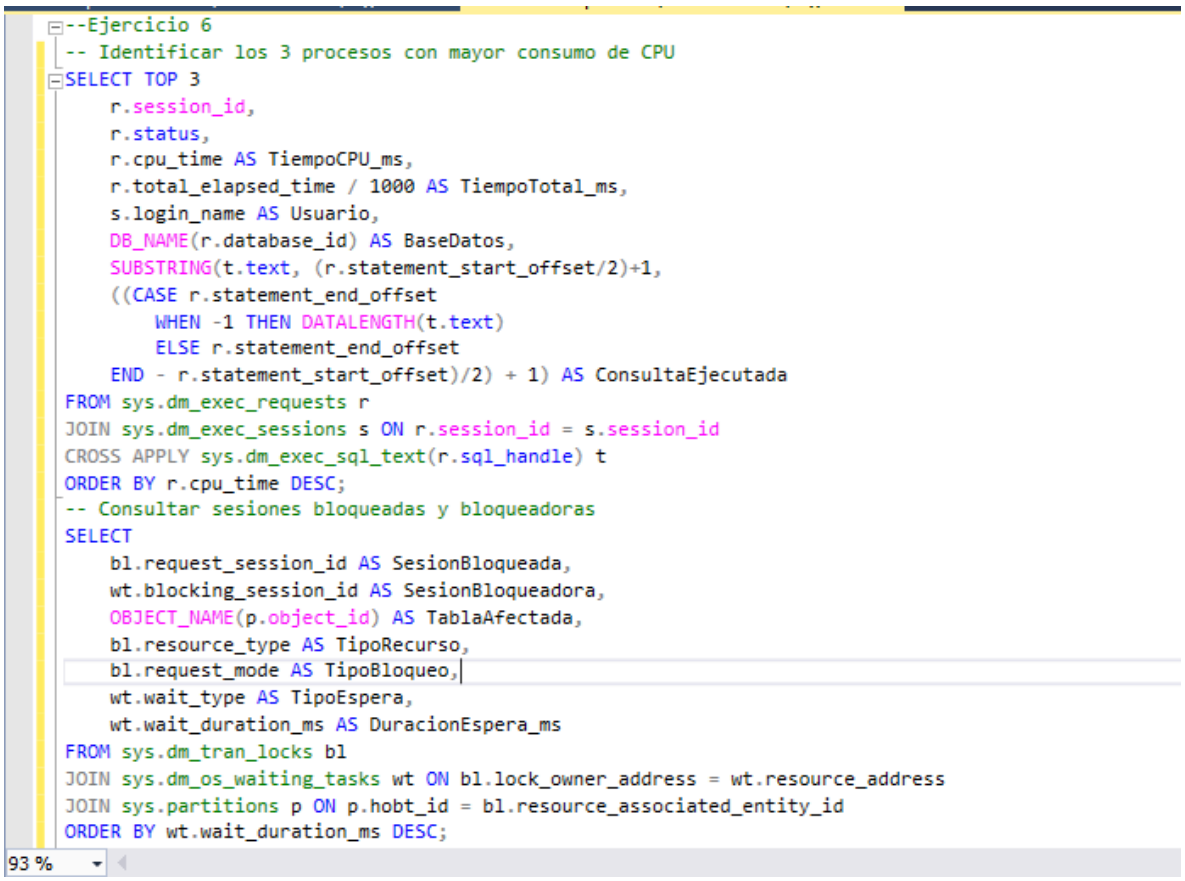
# Proyecto 6: Identificación y solución de procesos lentos en QhatuPeru

**1. Enunciado del ejercicio**

Durante una campaña promocional, el sistema **QhatuPeru** presenta lentitud en su desempeño general.
El equipo técnico debe identificar los **procesos con mayor consumo de CPU** y determinar si existen **bloqueos activos** que estén afectando el rendimiento del sistema.
Para ello se utilizarán herramientas de monitoreo como **Activity Monitor** y consultas en **T-SQL** que permitan analizar el estado actual del servidor SQL y sus sesiones activas.

**Ejercicio práctico:**
a) Utilizar **Activity Monitor** para identificar los tres procesos que consumen más CPU durante una operación masiva.
b) Ejecutar una consulta **T-SQL** para visualizar los bloqueos actuales en la base de datos.

**2. Script de la solución en T-SQL**



```sql
--Ejercicio 6
-- Identificar los 3 procesos con mayor consumo de CPU
SELECT TOP 3
    r.session_id,
    r.status,
    r.cpu_time AS TiempoCPU_ms,
    r.total_elapsed_time / 1000 AS TiempoTotal_ms,
    s.login_name AS Usuario,
    DB_NAME(r.database_id) AS BaseDatos,
    SUBSTRING(t.text, (r.statement_start_offset/2)+1,
    ((CASE r.statement_end_offset
        WHEN -1 THEN DATALENGTH(t.text)
        ELSE r.statement_end_offset
    END - r.statement_start_offset)/2) + 1) AS ConsultaEjecutada
FROM sys.dm_exec_requests r
JOIN sys.dm_exec_sessions s ON r.session_id = s.session_id
CROSS APPLY sys.dm_exec_sql_text(r.sql_handle) t
ORDER BY r.cpu_time DESC;
-- Consultar sesiones bloqueadas y bloqueadoras
SELECT
    bl.request_session_id AS SesionBloqueada,
    wt.blocking_session_id AS SesionBloqueadora,
    OBJECT_NAME(p.object_id) AS TablaAfectada,
    bl.resource_type AS TipoRecurso,
    bl.request_mode AS TipoBloqueo,
    wt.wait_type AS TipoEspera,
    wt.wait_duration_ms AS DuracionEspera_ms
FROM sys.dm_tran_locks bl
JOIN sys.dm_os_waiting_tasks wt ON bl.lock_owner_address = wt.resource_address
JOIN sys.partitions p ON p.hobt_id = bl.resource_associated_entity_id
ORDER BY wt.wait_duration_ms DESC;
```

93 %

Results | Messages

| | session_id | status | TiempoCPU_ms | TiempoTotal_ms | Usuario | BaseDatos | ConsultaEjecutada |
|---|---|---|---|---|---|---|---|
| 1 | 70 | running | 748 | 0 | Aurora\frank | QhatuPERU | SELECT TOP 3 r.session_id, r.statu... |

| SesionBloqueada | SesionBloqueadora | TablaAfectada | TipoRecurso | TipoBloqueo | TipoEspera | DuracionEspera_ms |
|---|---|---|---|---|---|---|

### 3. Justificación técnica de la solución aplicada

- Se utilizaron **vistas dinámicas de administración (DMVs)** como `sys.dm_exec_requests`, `sys.dm_exec_sessions`, `sys.dm_exec_sql_text` y `sys.dm_tran_locks`, que permiten acceder a **información en tiempo real** sobre los procesos en ejecución.

- El uso de `ORDER BY r.cpu_time DESC` permite identificar **los procesos que más CPU consumen**, lo cual es clave para priorizar optimizaciones.

- El cruce con `sys.dm_exec_sql_text` permite visualizar **la consulta exacta** que está consumiendo recursos.

- En el segundo script, las uniones entre `sys.dm_tran_locks`, `sys.dm_os_waiting_tasks` y `sys.partitions` permiten **detectar bloqueos y**

**recursos afectados** (tablas, sesiones involucradas y tipo de espera).

**4. Explicación de las buenas prácticas utilizadas**

- **Monitoreo proactivo**: Se aplicó análisis con DMVs y Activity Monitor para anticipar cuellos de botella antes de que afecten al usuario.
- **Consultas con TOP y filtros**: Evita sobrecargar el sistema al mostrar solo los procesos más relevantes.
- **Documentación clara**: Cada campo del SELECT está comentado y nombrado para facilitar su lectura y mantenimiento.
- **Uso de vistas del sistema**: Permite un diagnóstico **seguro y sin modificar datos** en la base.
- **Análisis por CPU y bloqueos**: Atiende tanto el rendimiento del servidor como la integridad de la concurrencia.

# Proyecto 7: Automatización de respaldos y limpieza del sistema QhatuPeru

**1. Enunciado del ejercicio**

Con el fin de garantizar la **disponibilidad, integridad y rendimiento** del sistema **QhatuPeru**, se solicita automatizar dos tareas administrativas fundamentales en el servidor SQL:

- La **generación diaria de respaldos automáticos** de la base de datos.
- La **limpieza semanal** de los registros antiguos en la tabla `Sesiones` (más de 15 días de antigüedad).

**Ejercicio práctico:**
a) Crear un **Job en SQL Server Agent** que realice un respaldo diario automático de la base de datos QhatuPeru.
b) Diseñar y programar un **Job semanal** que elimine los registros de la tabla `Sesiones` con más de 15 días de antigüedad.

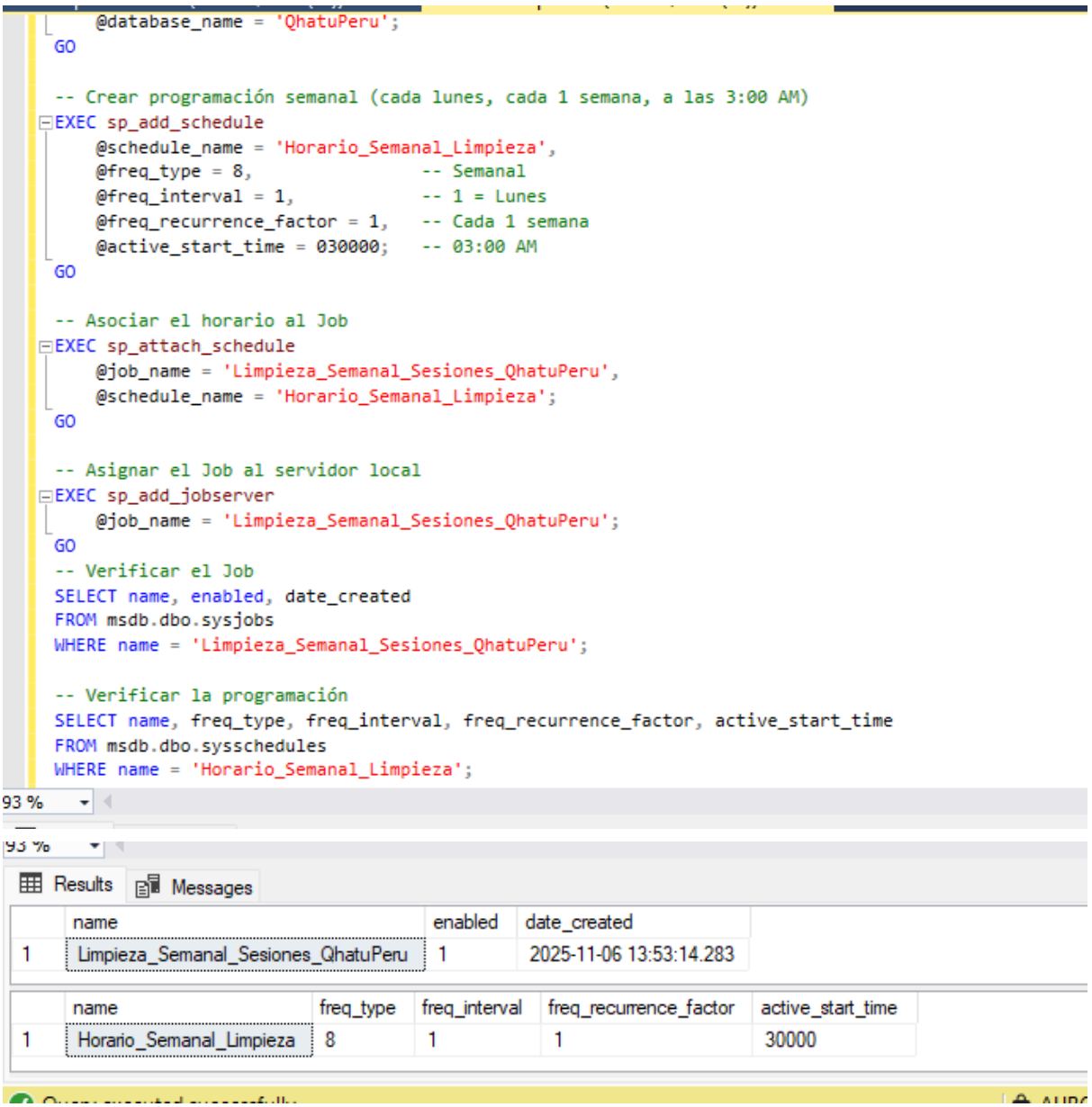
**2. Script de la solución en T-SQL**

```sql
--Ejercicio 7
-- Creación de Job para respaldo diario de la base QhatuPeru
USE msdb;
GO

EXEC sp_add_job
    @job_name = 'Backup_Diario_QhatuPeru',
    @description = 'Realiza un respaldo completo diario de la base QhatuPeru.';

-- Agregar paso al Job
EXEC sp_add_jobstep
    @job_name = 'Backup_Diario_QhatuPeru',
    @step_name = 'Paso_Respaldo',
    @subsystem = 'TSQL',
    @command = '
        BACKUP DATABASE QhatuPeru
        TO DISK = ''C:\Backups\QhatuPeru_Backup.bak''
        WITH INIT, NAME = ''Backup completo diario de QhatuPeru'',
        STATS = 10;',
    @database_name = 'master';

-- Crear programación diaria
EXEC sp_add_schedule
    @schedule_name = 'Horario_Diario_Backup',
    @freq_type = 4,              -- Diario
    @freq_interval = 1,          -- Cada 1 día
    @active_start_time = 020000; -- 2:00 AM
```

```sql
-- Asignar programación al Job
EXEC sp_attach_schedule
    @job_name = 'Backup_Diario_QhatuPeru',
    @schedule_name = 'Horario_Diario_Backup';

-- Asociar el Job al servidor
EXEC sp_add_jobserver
    @job_name = 'Backup_Diario_QhatuPeru';
GO
-- Creación de Job para limpieza semanal de sesiones antiguas
USE msdb;
GO

-- Crear el Job
EXEC sp_add_job
    @job_name = 'Limpieza_Semanal_Sesiones_QhatuPeru',
    @description = 'Elimina registros de la tabla Sesiones con más de 15 días de antigüedad.';
GO

-- Agregar el paso de limpieza
EXEC sp_add_jobstep
    @job_name = 'Limpieza_Semanal_Sesiones_QhatuPeru',
    @step_name = 'Paso_Limpieza',
    @subsystem = 'TSQL',
    @command = '
        USE QhatuPeru;
        DELETE FROM dbo.Sesiones
        WHERE FechaSesion < DATEADD(DAY, -15, GETDATE());
    ',
    @database_name = 'QhatuPeru';
```

93 %

Results Messages



```
    @database_name = 'QhatuPeru';
GO

-- Crear programación semanal (cada lunes, cada 1 semana, a las 3:00 AM)
EXEC sp_add_schedule
    @schedule_name = 'Horario_Semanal_Limpieza',
    @freq_type = 8,                -- Semanal
    @freq_interval = 1,            -- 1 = Lunes
    @freq_recurrence_factor = 1,   -- Cada 1 semana
    @active_start_time = 030000;   -- 03:00 AM
GO

-- Asociar el horario al Job
EXEC sp_attach_schedule
    @job_name = 'Limpieza_Semanal_Sesiones_QhatuPeru',
    @schedule_name = 'Horario_Semanal_Limpieza';
GO

-- Asignar el Job al servidor local
EXEC sp_add_jobserver
    @job_name = 'Limpieza_Semanal_Sesiones_QhatuPeru';
GO
-- Verificar el Job
SELECT name, enabled, date_created
FROM msdb.dbo.sysjobs
WHERE name = 'Limpieza_Semanal_Sesiones_QhatuPeru';

-- Verificar la programación
SELECT name, freq_type, freq_interval, freq_recurrence_factor, active_start_time
FROM msdb.dbo.sysschedules
WHERE name = 'Horario_Semanal_Limpieza';
```

93 %

93 %

Results | Messages

| | name | enabled | date_created |
|---|---|---|---|
| 1 | Limpieza_Semanal_Sesiones_QhatuPeru | 1 | 2025-11-06 13:53:14.283 |

| | name | freq_type | freq_interval | freq_recurrence_factor | active_start_time |
|---|---|---|---|---|---|
| 1 | Horario_Semanal_Limpieza | 8 | 1 | 1 | 30000 |

### 3. Justificación técnica de la solución aplicada

- Se emplearon los **procedimientos almacenados del sistema** de `msdb` (`sp_add_job`, `sp_add_jobstep`, `sp_add_schedule`, `sp_attach_schedule`, `sp_add_jobserver`) que permiten crear y automatizar tareas mediante el **SQL Server Agent**, garantizando ejecución automática sin intervención manual.
- El primer Job realiza un **respaldo completo diario** de la base de datos QhatuPeru, asegurando la recuperación ante fallos o pérdida de información.
  El segundo Job elimina periódicamente registros antiguos en la tabla `Sesiones`, reduciendo el tamaño de la base de datos y optimizando el rendimiento en consultas.
- Ambos procesos se ejecutan fuera del horario laboral (2:00 AM y 3:00 AM) para **minimizar el impacto en los usuarios**.

### 4. Explicación de las buenas prácticas utilizadas en el proyecto

- **Automatización programada:** evita tareas manuales y asegura la ejecución constante de respaldos y limpieza.

- **Separación de funciones:** los Jobs fueron diseñados de forma independiente para un mejor control y mantenimiento.

- **Ejecución fuera de horario pico:** se programaron horarios nocturnos para no afectar el rendimiento del sistema en uso.

- **Rutas claras y centralizadas:** los respaldos se almacenan en una carpeta específica (`C:\Backups\`) para facilitar su localización y gestión.

- **Uso de filtros temporales precisos:** la función `DATEADD(DAY, -15, GETDATE())` garantiza que solo se eliminen registros realmente antiguos, evitando pérdida de información reciente.

- **Registro descriptivo:** los nombres y descripciones de los Jobs son claros, facilitando su identificación en entornos de producción.

# Proyecto 8: Registro de estados en pedidos en QhatuPeru

**1. Enunciado del ejercicio**

Con la finalidad de mejorar la **trazabilidad y control de los envíos**, la empresa **QhatuPeru** requiere ajustar la estructura de la tabla `Pedidos`.
Se solicita agregar una nueva columna para priorizar los pedidos y, posteriormente, eliminar una columna que ya no será necesaria en el modelo de datos.

**Ejercicio práctico:**
a) Agregar la columna **“Prioridad”** de tipo **INT** a la tabla `Pedidos`.
b) Eliminar la columna **“EstadoEnvio”** de la tabla `Pedidos`.

**2. Script de la solución en T-SQL**

```sql
--Ejercicio 8
--a) Agregar la columna “Prioridad”

USE QhatuPERU;
GO

ALTER TABLE PEDIDOS
ADD Prioridad INT NULL;
GO
--b) Eliminar la columna “EstadoEnvio”
USE QhatuPERU;
GO

ALTER TABLE PEDIDOS
DROP COLUMN EstadoEnvio;
GO
```

```sql
-- RESULTADOS
USE QhatuPERU;
GO
SELECT name
FROM sys.columns
WHERE object_id = OBJECT_ID('dbo.PEDIDOS');

USE QhatuPERU;
GO
SELECT name
FROM sys.columns
WHERE object_id = OBJECT_ID('dbo.PEDIDOS')
  AND name = 'EstadoEnvio';
```

93 %

Results | Messages

| | name |
|---|---|
| 1 | IdPedido |
| 2 | IdCliente |
| 3 | FechaPedido |
| 4 | MontoTotal |
| 5 | Prioridad |

| | name |
|---|---|

### 3. Justificación técnica de la solución aplicada

Se empleó el comando **ALTER TABLE**, que permite modificar la estructura de una tabla existente sin necesidad de recrearla.

- En el primer caso, la instrucción ADD agrega la columna **Prioridad**, permitiendo registrar un valor numérico que indique la importancia o urgencia de un pedido (por

ejemplo, 1 = alta, 2 = media, 3 = baja).

- En el segundo caso, `DROP COLUMN` elimina la columna **EstadoEnvio**, que ya no es necesaria debido a una reestructuración o centralización del control de envíos.

Estas operaciones se realizan dentro del contexto de la base de datos **QhatuPeru**, asegurando que los cambios afecten únicamente a las tablas del sistema de gestión comercial correspondiente.

**4. Explicación de las buenas prácticas utilizadas en el proyecto**

- **Modificación controlada:** antes de aplicar los cambios, se recomienda verificar dependencias o restricciones sobre las columnas a modificar.

- **Uso del esquema explícito (dbo):** garantiza que las operaciones afecten la tabla correcta, incluso en entornos con múltiples esquemas.

- **Evitar pérdida de datos:** al agregar la columna `Prioridad` como `NULL`, se evita interrumpir operaciones existentes hasta que se definan valores válidos.

- **Documentación clara del cambio:** registrar la modificación en el control de versiones o bitácora técnica asegura trazabilidad en el mantenimiento del sistema.

- **Validación posterior:** tras aplicar los cambios, se puede usar `sp_help 'dbo.Pedidos'` o consultar `INFORMATION_SCHEMA.COLUMNS` para confirmar la estructura actualizada.

# Proyecto 9: Implementación de registros automáticos de modificaciones en QhatuPeru

## 1. Enunciado del ejercicio

Como parte de las políticas internas de seguridad y protección de datos de QhatuPeru, se requiere auditar las operaciones críticas realizadas sobre la tabla **Clientes**, en especial las eliminaciones de registros.
El sistema debe registrar automáticamente los datos eliminados para mantener un historial que permita rastrear cambios y responsables de dichas operaciones.

**Ejercicio práctico:**
a) Crear la tabla `AuditoriaClientes` para registrar cambios en `Clientes`.
b) Desarrollar un *trigger* que registre en `AuditoriaClientes` cada eliminación realizada en `Clientes`.

**2. Script de la solución en T-SQL**

```sql
--Ejercicio 9
USE QhatuPERU;
GO

-- a) Crear la tabla de auditoría
IF OBJECT_ID('dbo.AuditoriaClientes', 'U') IS NOT NULL
    DROP TABLE dbo.AuditoriaClientes;
GO

CREATE TABLE dbo.AuditoriaClientes (
    IdAuditoria INT IDENTITY(1,1) PRIMARY KEY,
    IdCliente INT NOT NULL,
    NombreCliente NVARCHAR(100),
    DNI CHAR(8),
    Telefono CHAR(9),
    Direccion NVARCHAR(150),
    Correo NVARCHAR(100),
    FechaEliminacion DATETIME DEFAULT GETDATE(),
    UsuarioEliminacion SYSNAME
);
GO

-- b) Crear el trigger para registrar eliminaciones
IF OBJECT_ID('dbo.tr_AuditarEliminacionClientes', 'TR') IS NOT NULL
    DROP TRIGGER dbo.tr_AuditarEliminacionClientes;
GO
```

```sql
-- b) Crear el trigger para registrar eliminaciones
IF OBJECT_ID('dbo.tr_AuditarEliminacionClientes', 'TR') IS NOT NULL
    DROP TRIGGER dbo.tr_AuditarEliminacionClientes;
GO

CREATE TRIGGER dbo.tr_AuditarEliminacionClientes
ON dbo.Clientes
AFTER DELETE
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO dbo.AuditoriaClientes (
        IdCliente, NombreCliente, DNI, Telefono, Direccion, Correo, FechaEliminacion, UsuarioEliminacion
    )
    SELECT
        d.IdCliente,
        d.NombreCliente,
        d.DNI,
        d.Telefono,
        d.Direccion,
        d.Correo,
        GETDATE(),
        SUSER_SNAME()
    FROM deleted d;
END;
GO
```

```sql
-- Prueba: eliminar un cliente de ejemplo
DELETE FROM dbo.Clientes WHERE IdCliente = 1;

-- Verificación: consultar la tabla de auditoría
SELECT * FROM dbo.AuditoriaClientes;
GO
```

93 %

Results | Messages

| IdAuditoria | IdCliente | NombreCliente | DNI | Telefono | Direccion | Correo | FechaEliminacion | UsuarioEliminacion |
|---|---|---|---|---|---|---|---|---|

## 3. Justificación técnica de la solución aplicada

- Se implementó la tabla AuditoriaClientes con un campo IdAuditoria autoincremental para mantener la trazabilidad de cada evento.

- Se incluyeron los campos de identificación (IdCliente, DNI) y de contacto (Teléfono, Dirección, Correo) para conservar información clave del cliente eliminado.

- El *trigger* `AFTER DELETE` utiliza la tabla lógica deleted, que contiene los registros antes de ser eliminados, y los inserta automáticamente en la tabla de auditoría.

- Se añadió el campo UsuarioEliminacion usando `SUSER_SNAME()` para identificar quién ejecutó la acción, y FechaEliminacion con `GETDATE()` para registrar el momento exacto.

**4. Explicación de las buenas prácticas utilizadas**

1. Separación de auditoría: los registros se almacenan en una tabla independiente para evitar sobrecargar la tabla principal.

2. Uso de triggers post-acción: el *trigger* `AFTER DELETE` garantiza que el evento de eliminación se complete antes de registrar el cambio.

3. Trazabilidad completa: incluye usuario y fecha de eliminación para cumplir con las normas de seguridad.

4. Validaciones previas: se verifica la existencia de objetos antes de crearlos (`OBJECT_ID`), evitando errores al ejecutar múltiples veces el script.

5. Nombres descriptivos: `tr_AuditarEliminacionClientes` y `AuditoriaClientes` reflejan claramente su propósito, mejorando la mantenibilidad.

# Proyecto 10: Simulación de restauración tras un incidente en QhatuPeru

## 1. Enunciado del ejercicio

Durante las operaciones diarias de QhatuPeru, un error humano provocó la eliminación accidental de datos en la tabla **Clientes**.
El equipo técnico debe restaurar la base de datos utilizando el respaldo más reciente y confirmar la recuperación exitosa de la información afectada.

**Ejercicio práctico:**
a) Simular la eliminación de registros de la tabla `Clientes` y restaurar la base de datos desde un respaldo.

b) Validar la correcta restauración de los datos mediante una consulta posterior a la recuperación.

## 2. Script de la solución en T-SQL

**Nota previa:**
Antes de ejecutar esta simulación, asegúrate de tener un *backup* reciente de la base de datos **QhatuPERU** (por ejemplo: **`C:\Backups\QhatuPERU.bak`),** esto ya desarrollado en el ejercicio 7, pero si puedes hacerlo manualmente con el siguiente Script.

```
--Si prefieres hacerlo por código, ejecuta esto:

USE master;
GO

-- Asegúrate de que la carpeta exista
EXEC xp_create_subdir 'C:\Backups';
GO

-- Crear respaldo inmediato
BACKUP DATABASE QhatuPeru
TO DISK = 'C:\Backups\QhatuPeru_Backup.bak'
WITH INIT, FORMAT, NAME = 'Backup manual de QhatuPeru';
GO
```

Luego ejecutamos nuestro codigo:

```sql
--luego ejecuta lo siguiente
--a) Simulación del incidente (eliminación total de CLIENTES)

USE QhatuPERU;
GO

-- Mostrar registros actuales
SELECT TOP 10 * FROM CLIENTES;

-- Simular el error humano: eliminación de todos los registros
DELETE FROM CLIENTES;
GO

-- Verificar que se eliminaron los datos
SELECT COUNT(*) AS RegistrosRestantes FROM CLIENTES;
GO

--b) Restauración desde el respaldo más reciente

USE master;
GO

-- Crear carpeta de respaldo si no existe
EXEC xp_create_subdir 'C:\Users\Public\Documents\';
GO

-- Crear respaldo manual previo (si no existía)
BACKUP DATABASE QhatuPERU
TO DISK = 'C:\Users\Public\Documents\QhatuPERU.bak'
WITH INIT, FORMAT, NAME = 'Backup manual de QhatuPERU', STATS = 10;
GO
```

```sql
-- Forzar modo de usuario único para restaurar
ALTER DATABASE QhatuPERU SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
GO

-- Restaurar la base de datos desde el archivo de respaldo
RESTORE DATABASE QhatuPERU
FROM DISK = 'C:\Users\Public\Documents\QhatuPERU.bak'
WITH REPLACE, RECOVERY, STATS = 5;
GO

-- Regresar la base a modo multiusuario
ALTER DATABASE QhatuPERU SET MULTI_USER;
GO

--Verificación de la restauración

USE QhatuPERU;
GO

-- Mostrar los primeros registros restaurados
SELECT TOP 10 * FROM CLIENTES;

-- Contar total de registros restaurados
SELECT COUNT(*) AS RegistrosRestaurados FROM CLIENTES;
GO
```


### 3. Justificación técnica de la solución aplicada

- Se utilizó el comando `BACKUP DATABASE` para generar un respaldo físico completo en una carpeta del sistema con permisos de lectura/escritura para SQL Server.

- El uso de `xp_create_subdir` asegura que la carpeta de destino exista antes del respaldo.

- Se empleó `ALTER DATABASE ... SET SINGLE_USER WITH ROLLBACK IMMEDIATE` para forzar el acceso exclusivo antes de la restauración y evitar conflictos con conexiones activas.

- El comando `RESTORE DATABASE` con las opciones `WITH REPLACE` y `RECOVERY` permite sobrescribir la base de datos dañada y devolverla a su estado operativo

normal.

- La verificación posterior mediante `SELECT` garantiza que los datos de la tabla *Clientes* fueron restaurados correctamente.

**4. Explicación de las buenas prácticas utilizadas en el proyecto**

1. **Separación de operaciones críticas:**
   Se ejecuta primero el respaldo, luego la simulación del error y finalmente la restauración, evitando pérdida accidental de datos.

2. **Rutas seguras y accesibles:**
   Se usa `C:\Users\Public\Documents\` para garantizar permisos adecuados del servicio SQL Server.

3. **Modo de usuario único:**
   Antes de restaurar, se cambia el estado de la base a *SINGLE_USER* para evitar bloqueos o accesos concurrentes.

4. **Restauración controlada:**
   Se utiliza `WITH REPLACE` para sobrescribir de forma segura y `STATS` para monitorear el progreso de la operación.

5. **Verificación post-restauración:**
   Se comprueba el número de registros restaurados y se visualizan los primeros resultados para validar integridad.

6. **Documentación clara y modular:**
   Cada bloque de código (respaldo, eliminación, restauración y verificación) está debidamente comentado y separado.