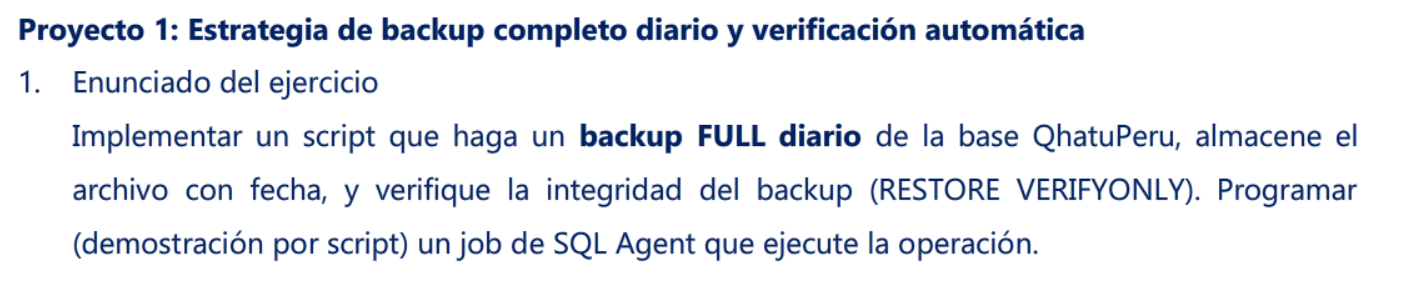
Actividad - Semana 12



Script:

-- Declarar variables para la ruta del backup y el nombre del archivo

DECLARE @RutaBackup NVARCHAR(500);

DECLARE @NombreArchivo NVARCHAR(100);

DECLARE @FechaActual NVARCHAR(20);

-- Definir la ruta base para guardar el backup.

-- ¡IMPORTANTE!: Reemplaza 'C:\BackupSQL\' con la ruta real en tu servidor.

SET @RutaBackup = 'C:\BackupSQL\';

-- Obtener la fecha y hora actual en un formato adecuado para el nombre del archivo (AAAA\_MM\_DD\_HHMMSS)

SELECT @FechaActual = CONVERT(NVARCHAR(20), GETDATE(), 112) + '\_' + REPLACE(CONVERT(NVARCHAR(8), GETDATE(), 108), ':', '');

-- Construir el nombre completo del archivo de backup

SET @NombreArchivo = @RutaBackup + 'QhatuPERU\_FULL\_' + @FechaActual + '.bak';

-- 1. Realizar el Backup FULL

BACKUP DATABASE [QhatuPERU]

TO DISK = @NombreArchivo

WITH

NOINIT, -- Añade el backup al medio sin sobrescribir si el archivo ya existe (aunque el nombre con fecha lo evita)

NAME = N'QhatuPERU Full Backup Diario', -- Descripción del backup

STATS = 10; -- Muestra el progreso en intervalos del 10%

-- 2. Verificar la Integridad del Backup (RESTORE VERIFYONLY)

RESTORE VERIFYONLY

FROM DISK = @NombreArchivo;

-- Mostrar un mensaje de éxito o fallo después de la verificación

IF @@ERROR = 0

BEGIN

PRINT 'El Backup de QhatuPERU se completó y su integridad ha sido VERIFICADA exitosamente.';

END

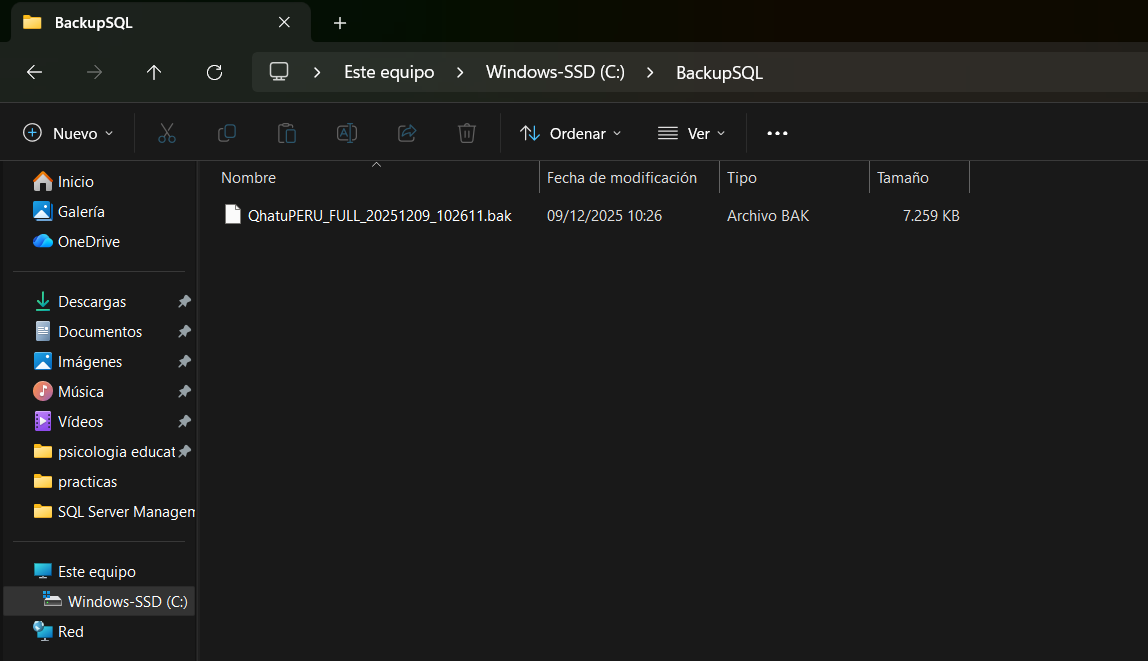
ELSE

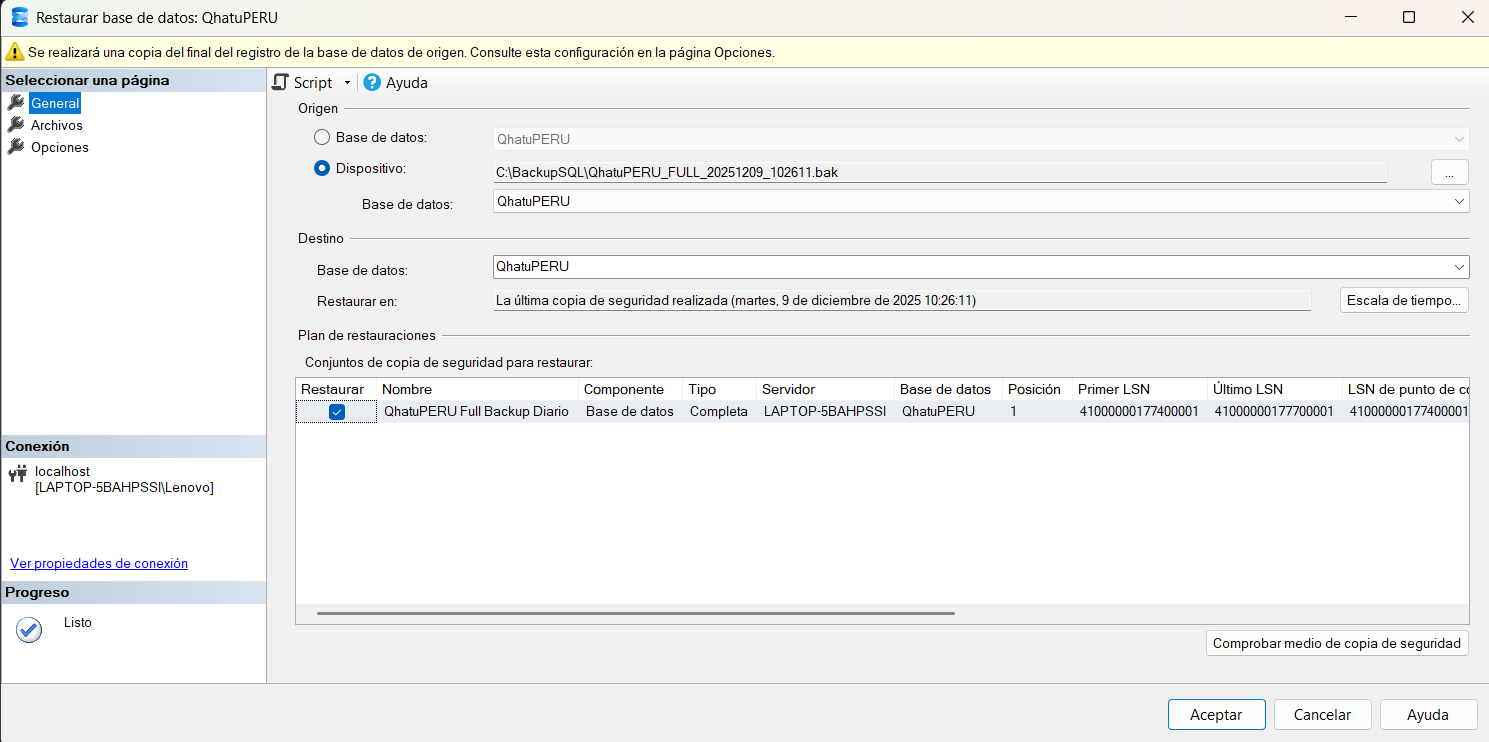
BEGIN

PRINT 'ERROR: Ocurrió un problema durante el proceso de Backup o su verificación.';

END

Justificación:

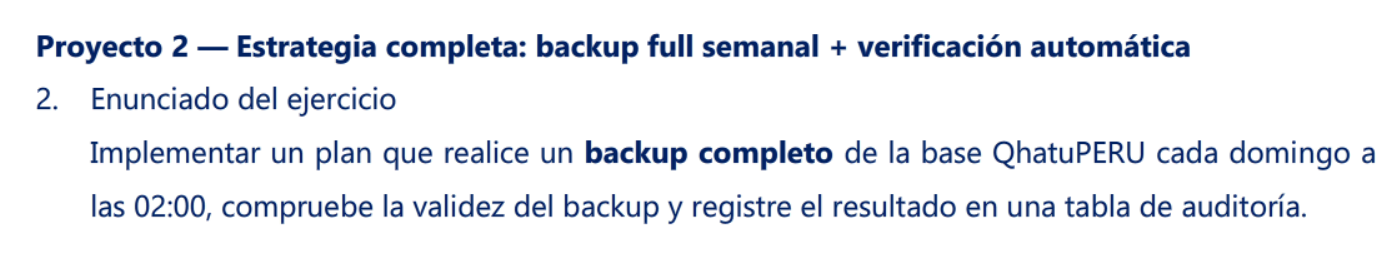




Buenas Prácticas:

La práctica más crítica es usar RESTORE VERIFYONLY justo después de crear el backup. Esto asegura que el archivo .bak se haya escrito correctamente en el disco y que es físicamente legible, previniendo la dolorosa sorpresa de descubrir que un backup está corrupto justo cuando se necesita.

Incluir la fecha y hora en el nombre del archivo (QhatuPERU\_FULL\_20251209\_...) evita la sobreescritura accidental y permite mantener un historial claro de backups.



Script:

Creacion de la tabla **auditoria:**

USE [QhatuPERU];

GO

IF OBJECT\_ID('dbo.Auditoria\_Backups') IS NULL

BEGIN

CREATE TABLE dbo.Auditoria\_Backups (

ID INT IDENTITY(1,1) PRIMARY KEY,

FechaEjecucion DATETIME NOT NULL,

TipoBackup VARCHAR(50) NOT NULL DEFAULT 'FULL Semanal',

BaseDatos VARCHAR(128) NOT NULL,

RutaArchivo NVARCHAR(500) NOT NULL,

EstadoValidacion VARCHAR(50) NOT NULL, -- 'OK' o 'FALLO'

MensajeError NVARCHAR(MAX) NULL

);

PRINT 'Tabla de Auditoría\_Backups creada exitosamente.';

END

ELSE

BEGIN

PRINT 'La tabla de Auditoría\_Backups ya existe.';

END

GO

Este script implementa toda la lógica: backup, verificación e inserción del resultado en la tabla de auditoría, manejando posibles errores.

-- Declaración de variables

DECLARE @RutaBackup NVARCHAR(500);

DECLARE @NombreArchivo NVARCHAR(200);

DECLARE @FechaActual NVARCHAR(20);

DECLARE @BaseDatos NVARCHAR(128) = N'QhatuPERU';

DECLARE @EstadoValidacion NVARCHAR(50);

DECLARE @MensajeError NVARCHAR(MAX) = NULL;

DECLARE @CodigoError INT;

-- Definir la ruta base para guardar el backup.

-- ¡IMPORTANTE!: Reemplaza 'C:\BackupSQL\' con tu ruta.

SET @RutaBackup = 'C:\BackupSQL\';

-- Obtener fecha y hora para el nombre del archivo

SELECT @FechaActual = CONVERT(NVARCHAR(20), GETDATE(), 112) + '\_' + REPLACE(CONVERT(NVARCHAR(8), GETDATE(), 108), ':', '');

SET @NombreArchivo = @RutaBackup + @BaseDatos + '\_FULL\_SEMANAL\_' + @FechaActual + '.bak';

-- Iniciar el bloque de Try-Catch para manejar errores

BEGIN TRY

-- 1. Realizar el Backup FULL

BACKUP DATABASE @BaseDatos

TO DISK = @NombreArchivo

WITH

NOINIT,

NAME = N'QhatuPERU Full Backup Semanal',

STATS = 10;

-- 2. Verificar la Integridad del Backup (RESTORE VERIFYONLY)

RESTORE VERIFYONLY

FROM DISK = @NombreArchivo;

-- Si no hay error, el estado es OK

SET @EstadoValidacion = 'OK';

END TRY

BEGIN CATCH

-- Si ocurre un error, capturamos la información

SET @EstadoValidacion = 'FALLO';

SELECT

@CodigoError = ERROR\_NUMBER(),

@MensajeError = 'Error Nro: ' + CONVERT(NVARCHAR(10), ERROR\_NUMBER()) +

', Mensaje: ' + ERROR\_MESSAGE() +

', Línea: ' + CONVERT(NVARCHAR(10), ERROR\_LINE());

END CATCH

-- 3. Registrar el Resultado en la Tabla de Auditoría

INSERT INTO [QhatuPERU].dbo.Auditoria\_Backups (

FechaEjecucion,

BaseDatos,

RutaArchivo,

EstadoValidacion,

MensajeError

)

VALUES (

GETDATE(),

@BaseDatos,

@NombreArchivo,

@EstadoValidacion,

@MensajeError

);

-- Mostrar mensaje final

IF @EstadoValidacion = 'OK'

BEGIN

PRINT 'Backup Semanal y Verificación de QhatuPERU completados y registrados como OK.';

END

ELSE

BEGIN

PRINT 'FALLO detectado. Revisar tabla de auditoría para detalles.';

END

Justificación:

Verificación mediante consulta

SELECT

FechaEjecucion,

TipoBackup,

BaseDatos,

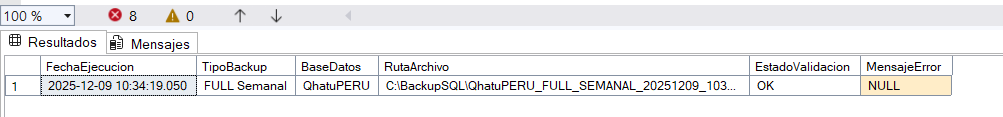
RutaArchivo,

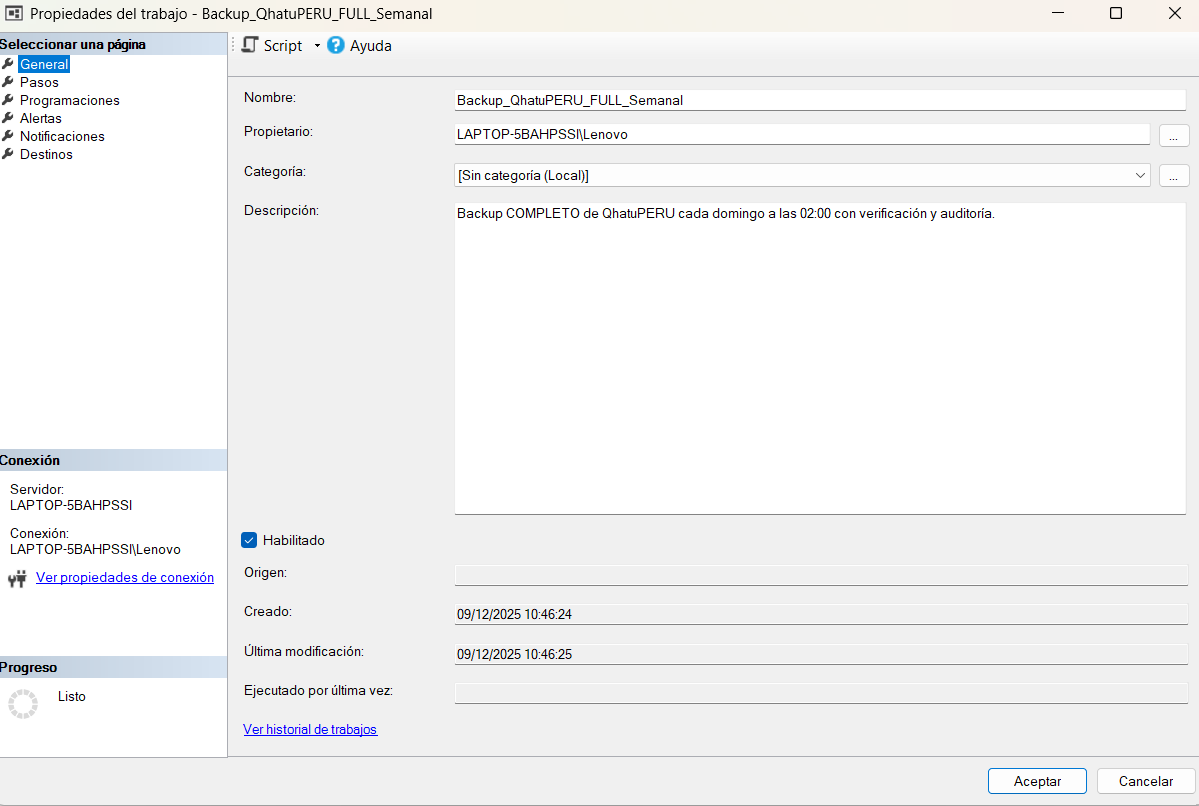
EstadoValidacion,

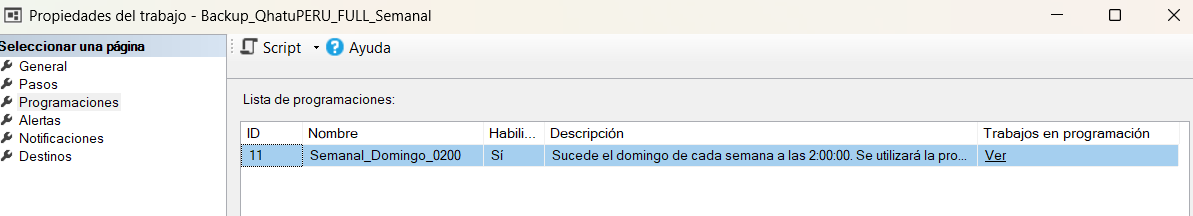
MensajeError

FROM [QhatuPERU].dbo.Auditoria\_Backups

ORDER BY FechaEjecucion DESC;







Buenas Prácticas:

* Verificación y Auditoría Centralizada: No solo se verifica el backup (RESTORE VERIFYONLY), sino que se registra el resultado de la verificación en una tabla de auditoría. Esta tabla es fundamental para que el DBA pueda revisar de forma rápida y centralizada el historial de éxito o fracaso de las copias de seguridad.
* Manejo de Errores Estructurado (TRY...CATCH): El script está envuelto en un bloque TRY...CATCH. Si el backup o la verificación fallan por cualquier motivo (ej. disco lleno, error de I/O), el bloque CATCH asegura que el error se capture, se almacene en la tabla de auditoría y se emita una alerta, previniendo fallos silenciosos.
* Frecuencia Adecuada: Establecer un backup FULL semanal proporciona un punto de recuperación de base conocido y consistente. La combinación con el backup diario del proyecto 1 (si se implementara) proporciona un modelo de recuperación robusto.



Script:

**Antes de realizar backups de transacciones (LOG), debemos asegurarnos de que la base de datos QhatuPERU esté en modo de recuperación FULL.**

ALTER DATABASE [QhatuPERU] SET RECOVERY FULL;

GO

Crearemos tres scripts diferentes, cada uno diseñado para ser ejecutado por un SQL Agent Job con distinta frecuencia.

**Backup FULL Semanal (FULL)**

Este script debe programarse para ejecutarse, por ejemplo, los **domingos a las 02:00:00**.

-- Script 1: Backup FULL Semanal

DECLARE @RutaFull NVARCHAR(500) = 'C:\BackupSQL\FULL\'; -- Carpeta dedicada para FULL

DECLARE @NombreArchivoFull NVARCHAR(200);

SELECT @NombreArchivoFull = @RutaFull + 'QhatuPERU\_FULL\_' +

REPLACE(CONVERT(NVARCHAR(10), GETDATE(), 120), '-', '') + '.bak';

BACKUP DATABASE [QhatuPERU]

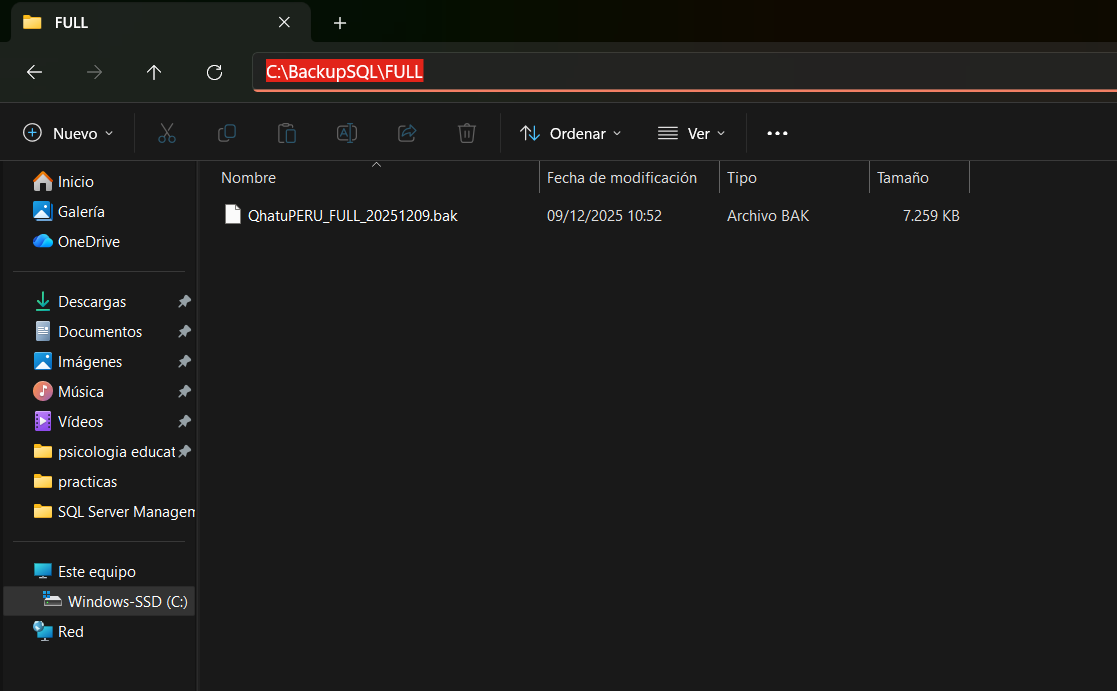
TO DISK = @NombreArchivoFull

WITH

INIT, -- Inicia un nuevo set de backups en el archivo

NAME = N'QhatuPERU Full Semanal',

STATS = 10;



**Backup Diferencial Diario (DIFF)**

Este script debe programarse para ejecutarse **diariamente** (de lunes a sábado), por ejemplo, a las **02:00:00**. Solo guarda los datos modificados desde el último backup **FULL**.

-- Script 2: Backup Diferencial Diario

DECLARE @RutaDiff NVARCHAR(500) = 'C:\BackupSQL\DIFF\'; -- Carpeta dedicada para DIFF

DECLARE @NombreArchivoDiff NVARCHAR(200);

SELECT @NombreArchivoDiff = @RutaDiff + 'QhatuPERU\_DIFF\_' +

REPLACE(CONVERT(NVARCHAR(10), GETDATE(), 120), '-', '') + '.dif';

BACKUP DATABASE [QhatuPERU]

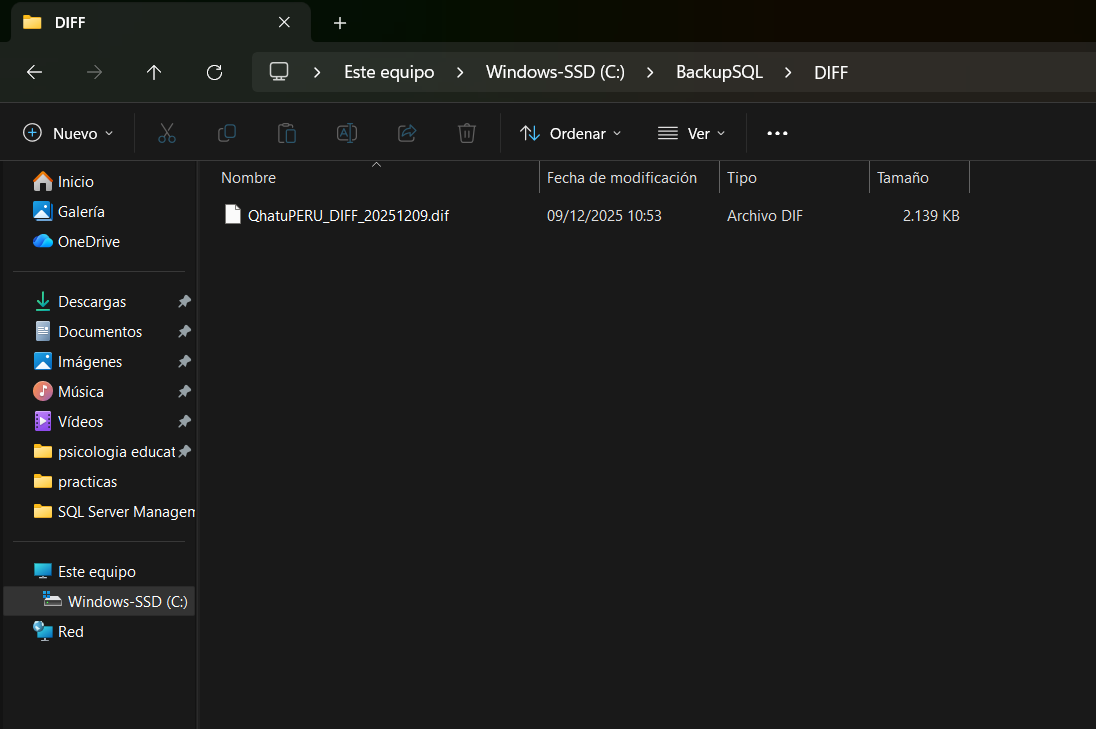
TO DISK = @NombreArchivoDiff

WITH

DIFFERENTIAL, -- ¡Clave! Indica que es un backup diferencial

NAME = N'QhatuPERU Diferencial Diario',

STATS = 10;



### Backup de Transacciones (LOG)

Este script debe programarse para ejecutarse **cada 30 minutos** (las 24 horas del día, los 7 días de la semana).

-- Script 3: Backup de Transacciones (LOG) - CORREGIDO

DECLARE @RutaLog NVARCHAR(500) = 'C:\practicas\bds\log\'; -- <<--- ¡Añadir el '\' al final de la ruta!

DECLARE @NombreArchivoLog NVARCHAR(200);

-- Se añade el nombre del archivo LOG

SELECT @NombreArchivoLog = @RutaLog + 'QhatuPERU\_Log\_' +

REPLACE(REPLACE(CONVERT(NVARCHAR(19), GETDATE(), 120), ' ', '\_'), ':', '') + '.trn';

-- Nota: También eliminé los dos puntos (:) del nombre para evitar futuros problemas en Windows

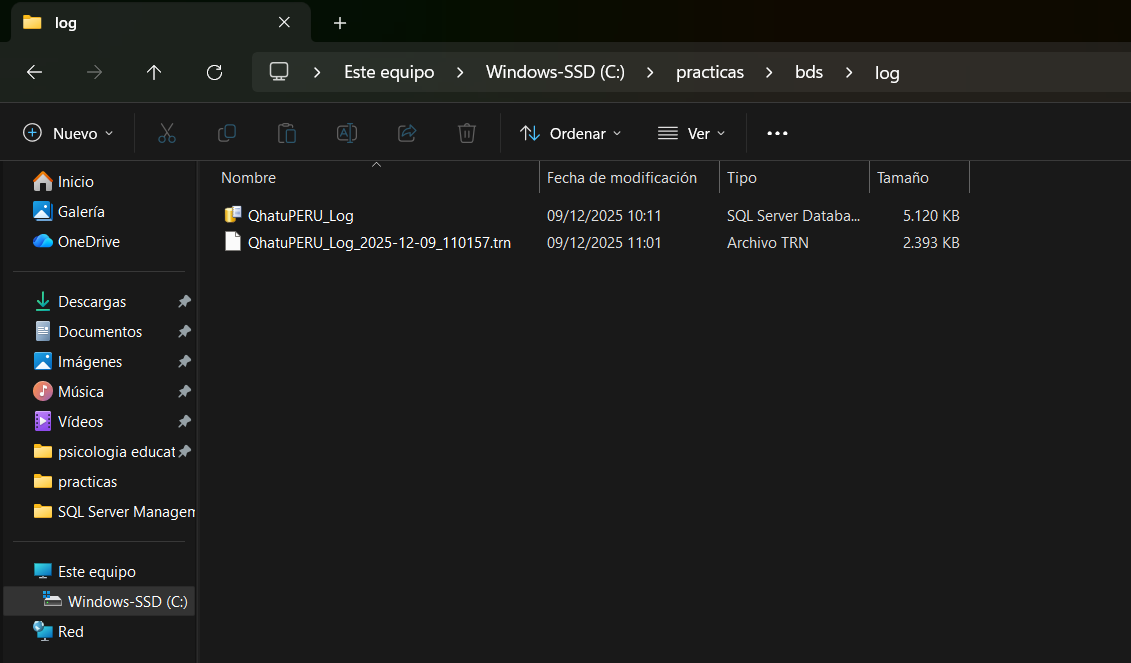
BACKUP LOG [QhatuPERU]

TO DISK = @NombreArchivoLog

WITH

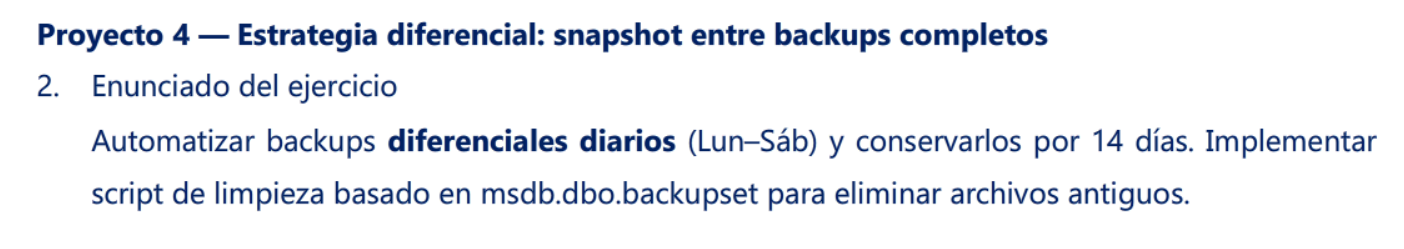
NOINIT,

NAME = N'QhatuPERU Log 30 Minutos';



Buenas Prácticas:

Restauración Rápida: Los backups diferenciales son más pequeños que los FULL y más rápidos de aplicar que aplicar muchos LOGs. Para restaurar, solo necesitas el FULL y el último DIFF, más los LOGs posteriores, lo que reduce significativamente el tiempo de recuperación (RTO).



Script:

## Script de Backup Diferencial Diario (Lunes–Sábado)

Este script debe programarse para ejecutarse **diariamente** (excepto domingo) a una hora fija (ej. 02:00:00).

-- Declarar variables

DECLARE @RutaDiff NVARCHAR(500) = 'C:\BackupSQL\DIFF\'; -- Asegúrate de que termine con '\'

DECLARE @NombreArchivoDiff NVARCHAR(200);

DECLARE @BaseDatos NVARCHAR(128) = N'QhatuPERU';

-- Generar nombre de archivo único

SELECT @NombreArchivoDiff = @RutaDiff + @BaseDatos + '\_DIFF\_' +

REPLACE(CONVERT(NVARCHAR(10), GETDATE(), 120), '-', '') + '.dif';

-- Realizar el Backup Diferencial

BACKUP DATABASE @BaseDatos

TO DISK = @NombreArchivoDiff

WITH

DIFFERENTIAL, -- ¡Clave! Indica que es un backup diferencial

NAME = N'QhatuPERU Diferencial Diario',

STATS = 10;

* Este script se configura en un SQL Agent Job con una recurrencia diaria, pero se debe excluir la ejecución del domingo (o programar para que se ejecute de Lunes a Sábado).

## Script de Limpieza de Archivos Antiguos (Basado en msdb.dbo.backupset)

Este script realiza dos tareas esenciales:

1. Identifica los registros de backups en la tabla del sistema msdb.dbo.backupset que tienen más de 14 días.
2. Elimina el archivo físico de backup correspondiente usando el comando xp\_cmdshell (requiere configuración) o el procedimiento sp\_delete\_backuphistory (el método más seguro).

### Pre-requisito: Habilitar xp\_cmdshell

La eliminación de archivos físicos requiere que el procedimiento xp\_cmdshell esté habilitado en SQL Server.

-- Ejecutar una sola vez para habilitar xp\_cmdshell

EXEC sp\_configure 'show advanced options', 1;

RECONFIGURE;

EXEC sp\_configure 'xp\_cmdshell', 1;

RECONFIGURE;

GO

### Script de Limpieza

Este script debe programarse para ejecutarse **diariamente** después del backup (ej. 03:00:00)

DECLARE @FechaLimite DATETIME;

DECLARE @ArchivoBackup NVARCHAR(512);

DECLARE @ComandoDel NVARCHAR(1000);

-- 1. Definir la fecha límite: 14 días antes de hoy

SET @FechaLimite = DATEADD(dd, -14, GETDATE());

PRINT '--- Iniciando limpieza de backups anteriores a: ' + CONVERT(NVARCHAR, @FechaLimite) + ' ---';

-- 2. Declarar cursor con el JOIN correcto para obtener la ruta física

DECLARE ArchivosCursor CURSOR LOCAL FOR

SELECT

bmf.physical\_device\_name

FROM msdb.dbo.backupset AS bs

INNER JOIN msdb.dbo.backupmediafamily AS bmf

ON bs.media\_set\_id = bmf.media\_set\_id

WHERE

bs.backup\_finish\_date < @FechaLimite

AND bs.database\_name = 'QhatuPERU'

AND bmf.physical\_device\_name NOT LIKE 'NULL' -- Evitar errores con backups virtuales

ORDER BY

bs.backup\_finish\_date;

OPEN ArchivosCursor;

FETCH NEXT FROM ArchivosCursor INTO @ArchivoBackup;

WHILE @@FETCH\_STATUS = 0

BEGIN

-- 3. Intentar eliminar el archivo físico del disco

IF @ArchivoBackup IS NOT NULL AND @ArchivoBackup != ''

BEGIN

-- Construir comando

SET @ComandoDel = N'DEL /F /Q "' + @ArchivoBackup + N'"';

-- Ejecutar comando

EXEC master..xp\_cmdshell @ComandoDel, NO\_OUTPUT;

PRINT 'Eliminando archivo: ' + @ArchivoBackup;

END

FETCH NEXT FROM ArchivosCursor INTO @ArchivoBackup;

END

CLOSE ArchivosCursor;

DEALLOCATE ArchivosCursor;

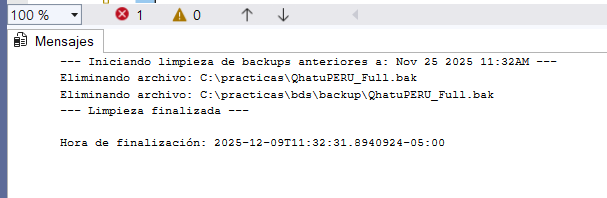
-- 4. Eliminar el historial de backups de MSDB

EXEC msdb.dbo.sp\_delete\_backuphistory @FechaLimite;

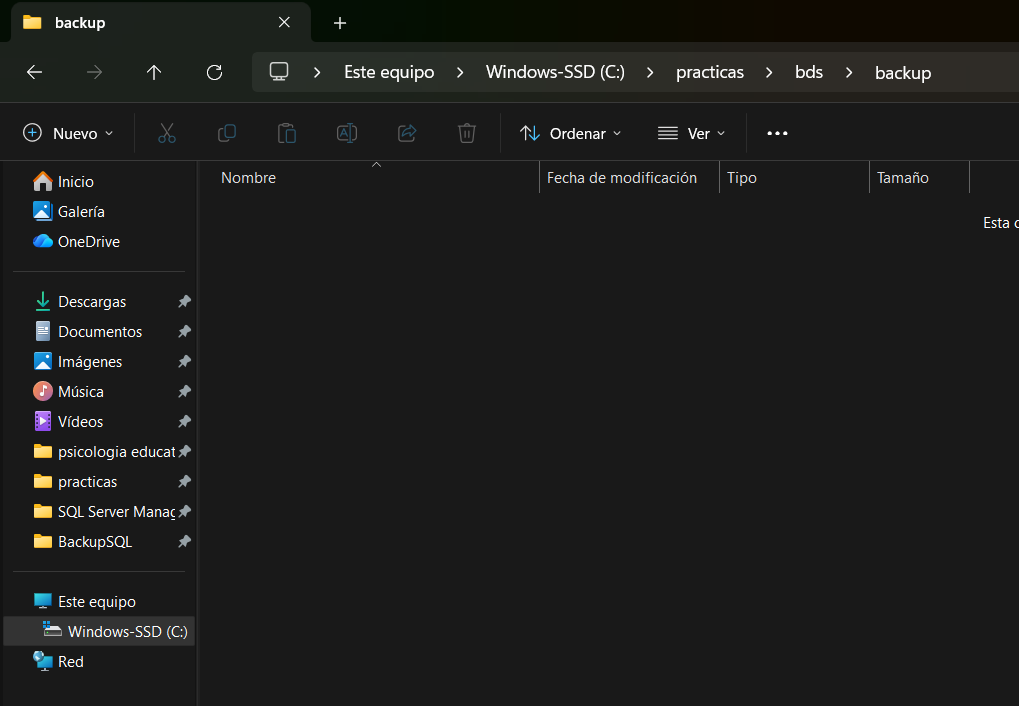
PRINT '--- Limpieza finalizada ---';

GO

Justificación:

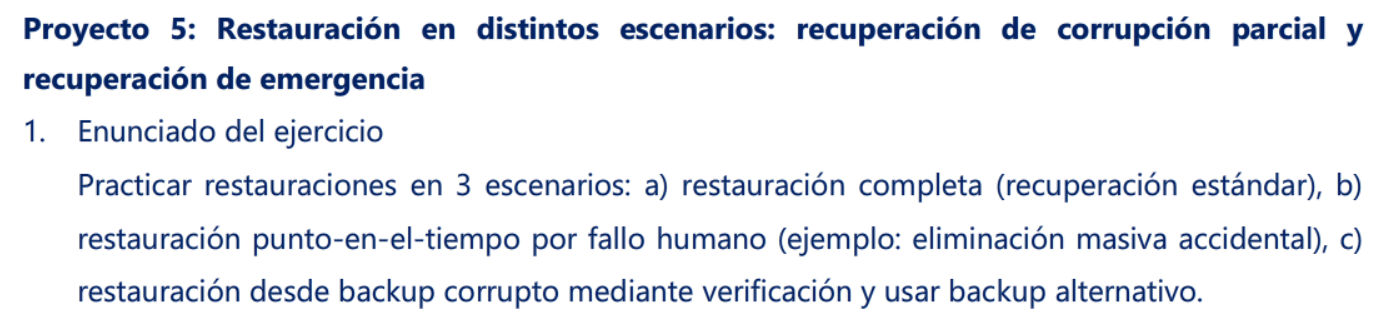


Se eliminaron archivos antiguos de mas de 14 dias



Buenas Prácticas:

La política de conservación de 14 días se logra al usar DATEADD(dd, -14, GETDATE()). Esto significa que, si el backup FULL es el domingo, los dos últimos backups FULLs siempre se mantendrán, al igual que los últimos 14 backups diferenciales.



Script:

### Escenario A: Restauración Completa (Estándar)

Este es el caso más común: el servidor falla y necesitas traer de vuelta la base de datos completa desde el último backup FULL válido.

USE master;

GO

-- 1. Asegurar que nadie esté usando la BD (poniéndola en modo SINGLE\_USER)

ALTER DATABASE [QhatuPERU] SET SINGLE\_USER WITH ROLLBACK IMMEDIATE;

-- 2. Restaurar el Backup FULL (Reemplaza la ruta con tu archivo real)

RESTORE DATABASE [QhatuPERU]

FROM DISK = N'C:\BackupSQL\QhatuPERU\_FULL\_SEMANAL\_20251209\_103418.bak'

WITH

REPLACE, -- Sobrescribe la base de datos existente

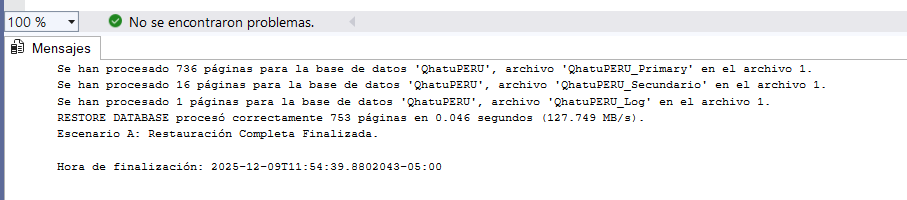
RECOVERY; -- Pone la base de datos en línea lista para usarse

-- 3. Devolver la BD a modo multi-usuario

ALTER DATABASE [QhatuPERU] SET MULTI\_USER;

GO

PRINT 'Escenario A: Restauración Completa Finalizada.';



### Escenario B: Recuperación "Point-in-Time" por Fallo Humano

**Situación:** Alguien borró datos importantes por accidente a las **10:00:00**. Tú tienes un backup FULL de las 08:00 y backups de LOG posteriores. Debes restaurar la base de datos tal como estaba a las **09:59:59**.  
USE master;

GO

-- 1. Crear un Backup FULL Base (Punto de partida)

BACKUP DATABASE [QhatuPERU]

TO DISK = N'C:\practicas\bds\full\QhatuPERU\_FULL\_Para\_EscenarioB.bak'

WITH FORMAT, NAME = 'Full Backup Base';

GO

-- Simular espera de 2 segundos

WAITFOR DELAY '00:00:02';

-- 2. Crear el Backup de LOG que nos faltaba (El archivo que "contiene" el tiempo)

BACKUP LOG [QhatuPERU]

TO DISK = N'C:\practicas\bds\log\QhatuPERU\_LOG\_Para\_EscenarioB.trn'

WITH FORMAT, NAME = 'Log Backup Base';

GO

PRINT '✅ Archivos creados correctamente. Ahora tienes el FULL y el LOG.';

Correr el escenario:  
USE master;

GO

PRINT '--- Iniciando Escenario B ---';

-- 1. Backup del "Tail-Log" (Cola del registro) por seguridad

-- Usamos TRY/CATCH por si la base de datos ya está en estado "Restoring" del intento fallido anterior

BEGIN TRY

BACKUP LOG [QhatuPERU]

TO DISK = N'C:\practicas\bds\log\QhatuPERU\_TailLog\_Emergencia.trn'

WITH NO\_TRUNCATE, INIT;

PRINT 'Tail-Log generado.';

END TRY

BEGIN CATCH

PRINT 'Aviso: No se pudo generar Tail-Log (la BD podría estar en uso o restaurando). Continuando...';

END CATCH

-- 2. Poner BD en modo exclusivo (Expulsar usuarios)

ALTER DATABASE [QhatuPERU] SET SINGLE\_USER WITH ROLLBACK IMMEDIATE;

-- 3. Restaurar el FULL que creamos en el Paso 1

-- IMPORTANTE: NORECOVERY para permitir aplicar el Log después

RESTORE DATABASE [QhatuPERU]

FROM DISK = N'C:\practicas\bds\full\QhatuPERU\_FULL\_Para\_EscenarioB.bak'

WITH REPLACE, NORECOVERY;

PRINT 'Backup FULL restaurado (en espera de logs)...';

-- 4. Restaurar el LOG que creamos en el Paso 1

-- Usamos una fecha dinámica para el STOPAT (Fecha actual + 1 hora) para asegurar que incluya todo lo recién creado

DECLARE @MomentoRecuperacion DATETIME = DATEADD(hour, 1, GETDATE());

RESTORE LOG [QhatuPERU]

FROM DISK = N'C:\practicas\bds\log\QhatuPERU\_LOG\_Para\_EscenarioB.trn'

WITH

STOPAT = @MomentoRecuperacion, -- Recuperar hasta este momento

RECOVERY; -- ¡AQUÍ TERMINAMOS Y PONEMOS LA BD ONLINE!

PRINT 'Backup LOG restaurado exitosamente.';

-- 5. Regresar a modo multi-usuario

ALTER DATABASE [QhatuPERU] SET MULTI\_USER;

GO

PRINT '✅ Escenario B: Restauración Point-in-Time COMPLETADA CORRECTAMENTE.';  
  
Si marca error, reinicia para poder realizar cambios a la base de datos y no se quede en fase de restauración:  
USE master;

GO

PRINT '--- Iniciando Reinicio de QhatuPERU ---';

-- 1. Intentar poner la base de datos en línea (Abortar la restauración pendiente)

BEGIN TRY

RESTORE DATABASE [QhatuPERU] WITH RECOVERY;

PRINT '✅ Se logró recuperar la base de datos existente.';

END TRY

BEGIN CATCH

PRINT '⚠️ No se pudo recuperar (faltan archivos). Procediendo a reiniciar desde cero...';

-- 2. Si no se puede recuperar, la borramos forzosamente y creamos una nueva

IF EXISTS (SELECT name FROM sys.databases WHERE name = N'QhatuPERU')

BEGIN

-- Poner en modo SINGLE\_USER para cerrar cualquier conexión y borrar

ALTER DATABASE [QhatuPERU] SET SINGLE\_USER WITH ROLLBACK IMMEDIATE;

DROP DATABASE [QhatuPERU];

PRINT '🗑️ Base de datos trabada eliminada.';

END

-- 3. Crear una base de datos limpia para el ejercicio

CREATE DATABASE [QhatuPERU];

PRINT '✨ Nueva base de datos QhatuPERU creada correctamente.';

END CATCH

GO

### Escenario C: Restauración desde Backup Corrupto (Simulación)

**Situación:** Intentas restaurar el backup principal, pero está dañado. El script debe detectar el fallo y automáticamente intentar usar un backup alternativo (uno anterior).

USE master;

GO

-- Variables para rutas

DECLARE @BackupPrincipal NVARCHAR(200) = 'C:\practicas\bds\full\QhatuPERU\_FULL\_Principal.bak';

DECLARE @BackupAlternativo NVARCHAR(200) = 'C:\practicas\bds\full\QhatuPERU\_FULL\_Alternativo.bak';

PRINT 'Iniciando verificación del Backup Principal...';

BEGIN TRY

-- 1. Intentar Verificar el Backup Principal

RESTORE VERIFYONLY FROM DISK = @BackupPrincipal;

-- Si pasa la línea anterior sin error, procedemos a restaurar el Principal

PRINT 'Backup Principal VERIFICADO. Restaurando...';

ALTER DATABASE [QhatuPERU] SET SINGLE\_USER WITH ROLLBACK IMMEDIATE;

RESTORE DATABASE [QhatuPERU] FROM DISK = @BackupPrincipal WITH REPLACE, RECOVERY;

ALTER DATABASE [QhatuPERU] SET MULTI\_USER;

PRINT 'Restauración exitosa desde el Principal.';

END TRY

BEGIN CATCH

-- 2. Si ocurre un error (Backup corrupto), el bloque CATCH toma el control

PRINT '⚠️ ALERTA: El Backup Principal está CORRUPTO o inaccesible.';

PRINT 'Error: ' + ERROR\_MESSAGE();

PRINT '🔄 Intentando restaurar desde el Backup ALTERNATIVO...';

BEGIN TRY

ALTER DATABASE [QhatuPERU] SET SINGLE\_USER WITH ROLLBACK IMMEDIATE;

RESTORE DATABASE [QhatuPERU] FROM DISK = @BackupAlternativo WITH REPLACE, RECOVERY;

ALTER DATABASE [QhatuPERU] SET MULTI\_USER;

PRINT '✅ Restauración de emergencia exitosa desde el Alternativo.';

END TRY

BEGIN CATCH

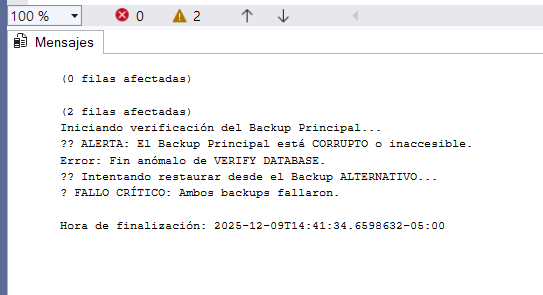
PRINT '❌ FALLO CRÍTICO: Ambos backups fallaron.';

ALTER DATABASE [QhatuPERU] SET MULTI\_USER;

END CATCH

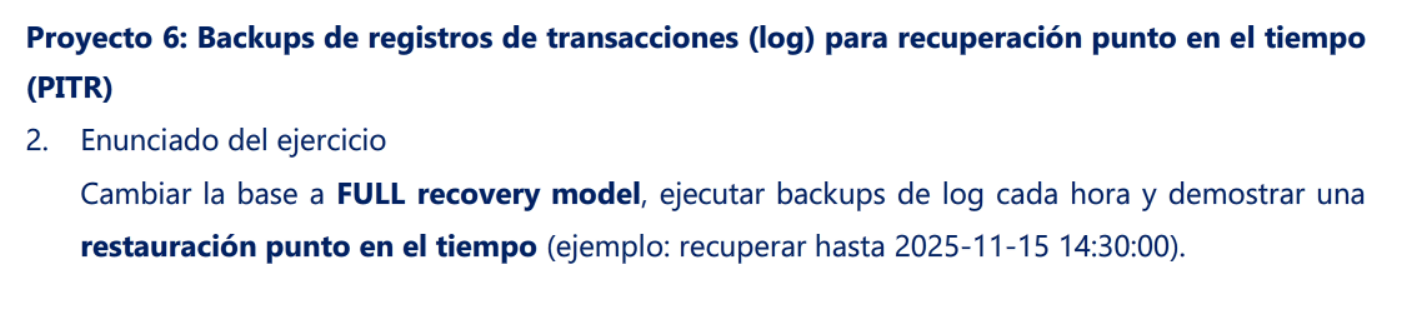
END CATCH;

GO



Buenas Prácticas:

* STOPAT (Point-in-Time Recovery): Es la herramienta más potente para corregir errores humanos sin perder todo el trabajo del día, sino solo lo ocurrido después del error.
* Tail-Log Backup: Antes de cualquier restauración crítica, siempre se debe intentar hacer un backup del log actual (NO\_TRUNCATE). Esto asegura que, si la restauración falla o nos arrepentimos, tenemos los datos "vivos" guardados hasta el último segundo antes de tocar nada.



Script:

Como no podemos esperar horas reales, ejecutaremos un script que **simulará** el paso del tiempo y la creación de datos.

Este script hará lo siguiente automáticamente:

1. Configura la BD en modelo **FULL**.
2. Hace un **Backup Full** inicial.
3. Inserta un **Dato Correcto** (el que queremos salvar).
4. Te mostrará la **HORA EXACTA** que debes guardar.
5. Simula un **Error Humano** (inserta un dato malo).
6. Hace un **Backup del Log** final.

USE master;

GO

-- 1. Configurar Recovery Model a FULL

ALTER DATABASE [QhatuPERU] SET RECOVERY FULL;

GO

PRINT '--- Iniciando Simulación ---';

-- 2. Limpieza y preparación de tabla de prueba

USE [QhatuPERU];

IF OBJECT\_ID('dbo.VentasDemo') IS NOT NULL DROP TABLE dbo.VentasDemo;

CREATE TABLE dbo.VentasDemo (ID INT IDENTITY, Evento VARCHAR(100), HoraRegistro DATETIME DEFAULT GETDATE());

-- Insertamos dato inicial

INSERT INTO dbo.VentasDemo (Evento) VALUES ('Inicio de operaciones');

-- 3. BACKUP FULL (Punto de partida obligatorio)

BACKUP DATABASE [QhatuPERU]

TO DISK = 'C:\practicas\bds\full\QhatuPERU\_PITR\_Full.bak'

WITH FORMAT, NAME = 'Full Backup Base';

PRINT '✅ Backup FULL generado.';

-- Simulamos paso del tiempo...

WAITFOR DELAY '00:00:02';

-- 4. EL MOMENTO CLAVE (Dato que queremos recuperar)

INSERT INTO dbo.VentasDemo (Evento) VALUES ('Venta Importante - ESTE DATO LO QUEREMOS');

DECLARE @HoraObjetivo DATETIME = GETDATE(); -- Capturamos la hora exacta de este evento

-- Imprimimos la hora para que la copies

PRINT '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*';

PRINT 'COPIA ESTA FECHA Y HORA EXACTA PARA EL PASO 2:';

PRINT CONVERT(VARCHAR(30), @HoraObjetivo, 126); -- Formato ISO8601 con la 'T'

PRINT '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*';

-- Simulamos más tiempo...

WAITFOR DELAY '00:00:02';

-- 5. EL DESASTRE (Error humano)

INSERT INTO dbo.VentasDemo (Evento) VALUES ('ERROR: Venta duplicada accidentalmente - NO QUEREMOS ESTO');

PRINT '❌ Error humano simulado (Dato malo insertado).';

-- 6. BACKUP LOG (Captura todo, incluido el error)

-- En la vida real, esto sería tu backup de log programado de "la siguiente hora"

BACKUP LOG [QhatuPERU]

TO DISK = 'C:\practicas\bds\log\QhatuPERU\_PITR\_LogFinal.trn'

WITH FORMAT, NAME = 'Log con el Desastre';

PRINT '✅ Simulación terminada. Revisa la pestaña "Mensajes" para ver la hora.';

GO

2025-12-09T14:47:08.050

### Ejecutar la Restauración Punto en el Tiempo (PITR)

Ahora que tenemos los backups, vamos a restaurar la base de datos para que contenga la "Venta Importante" pero **elimine** el "Error humano".

USE master;

GO

-- 1. Expulsar usuarios para poder restaurar

ALTER DATABASE [QhatuPERU] SET SINGLE\_USER WITH ROLLBACK IMMEDIATE;

-- DEFINE AQUÍ TU HORA OBJETIVO (La que copiaste del paso anterior)

-- Ejemplo: SET @HoraRestauracion = '2025-12-09T15:30:05.450';

DECLARE @HoraRestauracion DATETIME;

SET @HoraRestauracion = '2025-12-09T14:47:08.050'; -- <<--- ¡EDITA ESTO!

PRINT '--- Restaurando hasta: ' + CONVERT(VARCHAR, @HoraRestauracion, 120) + ' ---';

-- 2. Restaurar el FULL (Con NORECOVERY para seguir agregando logs)

RESTORE DATABASE [QhatuPERU]

FROM DISK = 'C:\practicas\bds\full\QhatuPERU\_PITR\_Full.bak'

WITH REPLACE, NORECOVERY;

-- 3. Restaurar el LOG con STOPAT (La magia del PITR)

-- Le decimos: "Aplica los cambios, pero DETENTE justo en esta hora"

RESTORE LOG [QhatuPERU]

FROM DISK = 'C:\practicas\bds\log\QhatuPERU\_PITR\_LogFinal.trn'

WITH

STOPAT = @HoraRestauracion, -- Aquí es donde ocurre el viaje en el tiempo

RECOVERY; -- Finalizamos y ponemos online

-- 4. Volver a multi-usuario

ALTER DATABASE [QhatuPERU] SET MULTI\_USER;

GO

-- 5. Verificación Final

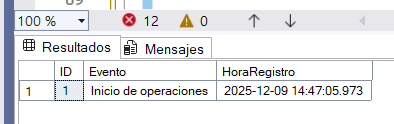
USE [QhatuPERU];

SELECT \* FROM dbo.VentasDemo;

-- Deberías ver la "Venta Importante" pero NO el "ERROR".

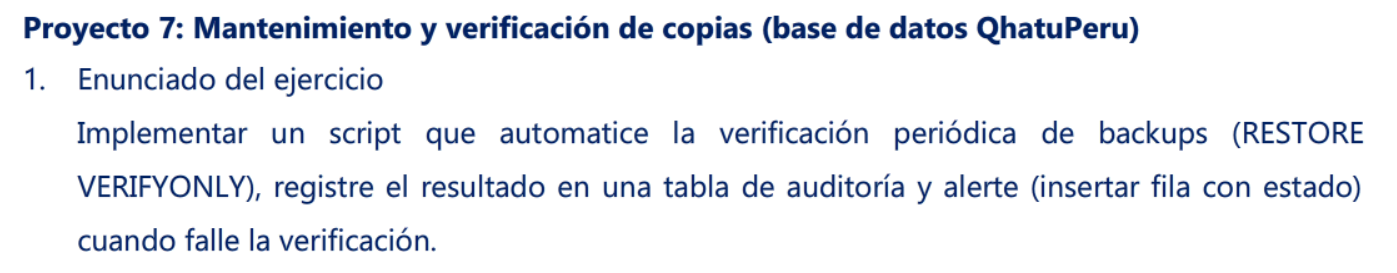
GO

Justificación:



Buenas Prácticas:

Las buenas prácticas implementadas en este procedimiento se centran en el uso del modelo de recuperación FULL, indispensable para habilitar la cadena de registros de transacciones y permitir la recuperación ante desastres con mínima pérdida de datos (RPO bajo). Es fundamental realizar un Backup Full inicial inmediatamente después de configurar este modelo para iniciar la cadena de logs, y posteriormente utilizar la cláusula STOPAT durante la restauración, lo que garantiza una recuperación granular (Point-in-Time Recovery) capaz de revertir errores humanos con precisión de segundos, gestionando siempre el acceso exclusivo (SINGLE\_USER) y los estados de recuperación (NORECOVERY/RECOVERY) para asegurar la integridad y consistencia de la base de datos durante todo el proceso.



Script:

## Asegurar la Tabla de Auditoría

Usaremos la misma tabla Auditoria\_Backups que creamos en el Proyecto 2. Si ya la tienes, este paso no hará nada; si la borraste, la volverá a crear.

USE [QhatuPERU];

GO

IF OBJECT\_ID('dbo.Auditoria\_Backups') IS NULL

BEGIN

CREATE TABLE dbo.Auditoria\_Backups (

ID INT IDENTITY(1,1) PRIMARY KEY,

FechaEjecucion DATETIME NOT NULL,

TipoBackup VARCHAR(50) NOT NULL,

BaseDatos VARCHAR(128) NOT NULL,

RutaArchivo NVARCHAR(500) NOT NULL,

EstadoValidacion VARCHAR(50) NOT NULL, -- 'OK' o 'FALLO'

MensajeError NVARCHAR(MAX) NULL

);

PRINT 'Tabla de Auditoría creada.';

END



## Script de Verificación Automática (El Robot)

Este script busca automáticamente el **último backup realizado** (sea Full, Diff o Log) en el historial de SQL Server, intenta leerlo (VERIFYONLY) y registra si está sano o corrupto.

DECLARE @UltimoBackup NVARCHAR(500);

DECLARE @Mensaje NVARCHAR(MAX);

DECLARE @Estado NVARCHAR(50);

-- 1. Buscar la ruta física del ÚLTIMO backup registrado en MSDB para QhatuPERU

SELECT TOP 1

@UltimoBackup = bmf.physical\_device\_name

FROM msdb.dbo.backupset AS bs

INNER JOIN msdb.dbo.backupmediafamily AS bmf

ON bs.media\_set\_id = bmf.media\_set\_id

WHERE bs.database\_name = 'QhatuPERU'

ORDER BY bs.backup\_finish\_date DESC; -- Ordenamos por fecha para obtener el más reciente

IF @UltimoBackup IS NOT NULL

BEGIN

PRINT 'Verificando archivo: ' + @UltimoBackup;

BEGIN TRY

-- 2. Ejecutar la verificación de integridad

RESTORE VERIFYONLY FROM DISK = @UltimoBackup;

-- Si llega aquí, es que todo salió bien

SET @Estado = 'EXITO';

SET @Mensaje = 'Verificación automática completada correctamente.';

END TRY

BEGIN CATCH

-- 3. Si falla, capturamos el error (¡Alerta!)

SET @Estado = 'FALLO';

SET @Mensaje = ERROR\_MESSAGE();

PRINT '¡ALERTA! El backup está corrupto.';

END CATCH

-- 4. Registrar el resultado en la tabla de auditoría

INSERT INTO [QhatuPERU].dbo.Auditoria\_Backups

(FechaEjecucion, TipoBackup, BaseDatos, RutaArchivo, EstadoValidacion, MensajeError)

VALUES

(GETDATE(), 'Verificacion Automatica', 'QhatuPERU', @UltimoBackup, @Estado, @Mensaje);

PRINT 'Resultado registrado en auditoría: ' + @Estado;

END

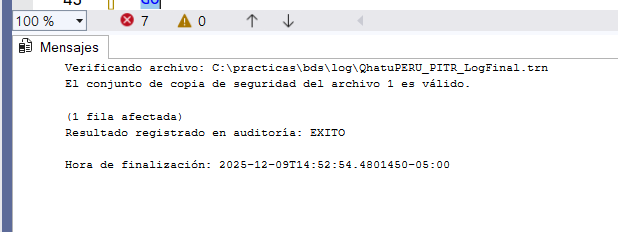
ELSE

BEGIN

PRINT 'No se encontraron backups recientes para verificar en el historial.';

END

GO



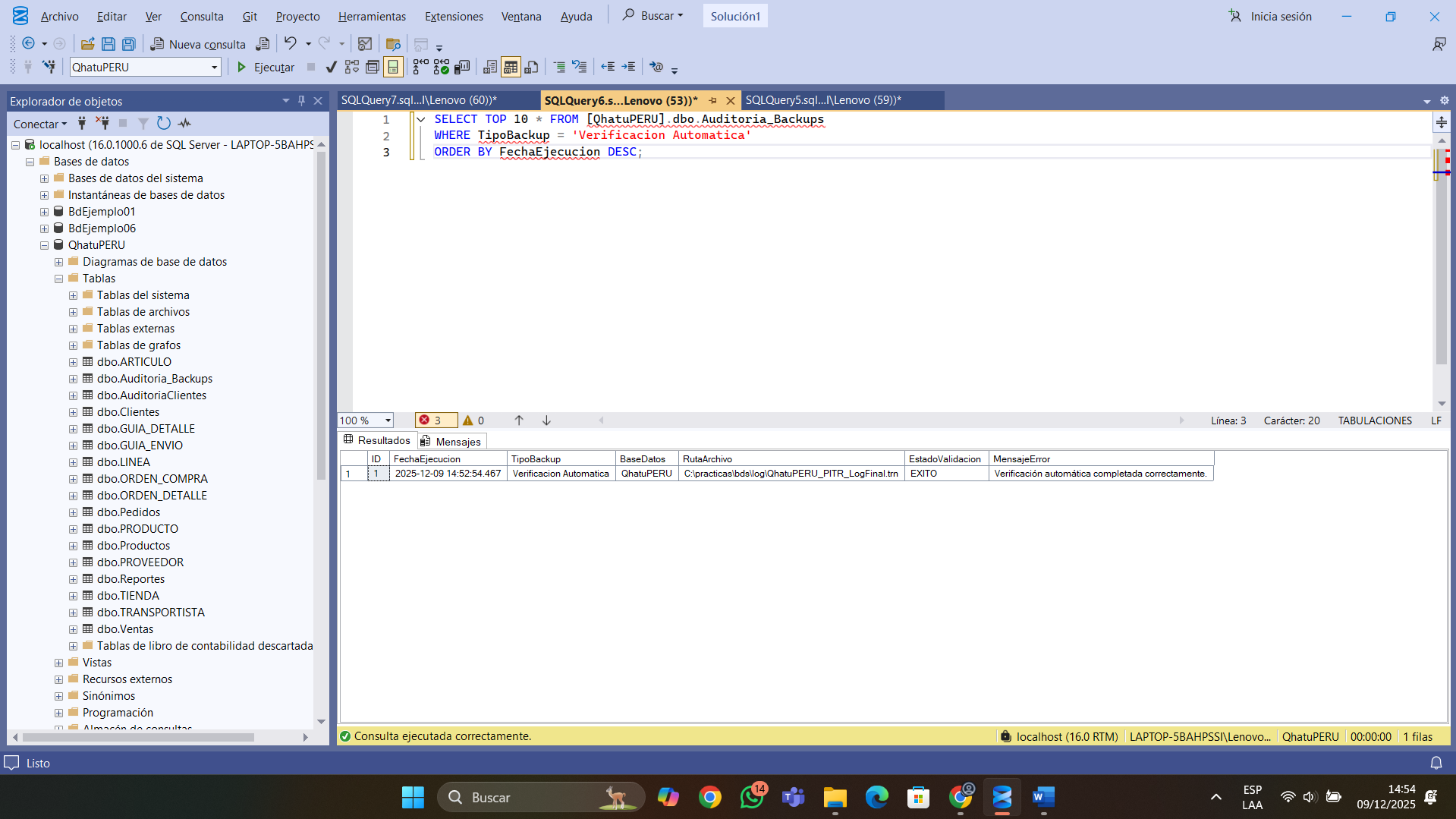
Para cumplir con el requisito de "verificación periódica", este script debería colocarse en un SQL Agent Job que se ejecute, por ejemplo, todos los días a las 06:00 AM (en un horario donde no choque con los backups).

Justificación:

SELECT TOP 10 \* FROM [QhatuPERU].dbo.Auditoria\_Backups

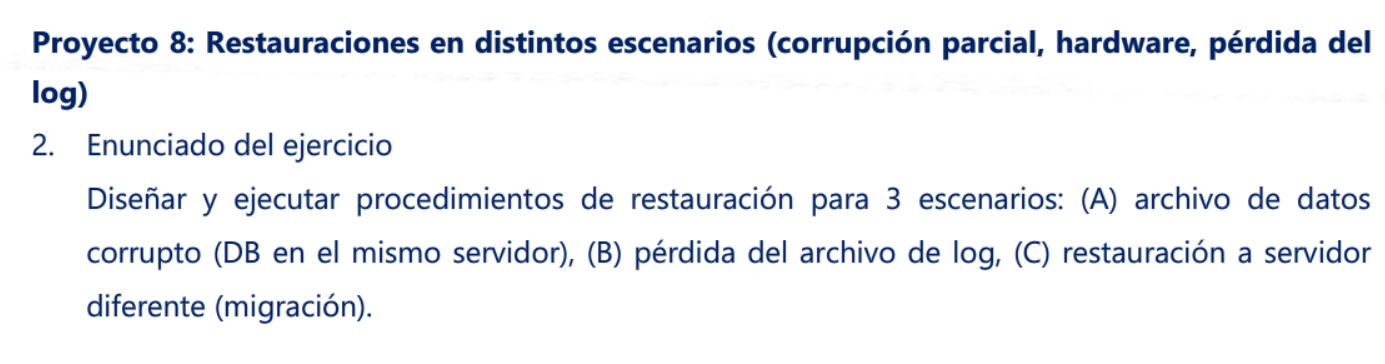
WHERE TipoBackup = 'Verificacion Automatica'

ORDER BY FechaEjecucion DESC;



Buenas Prácticas:

Este procedimiento implementa la Validación Proactiva de Recuperabilidad. En lugar de asumir que los backups funcionan solo porque el trabajo de backup terminó sin errores, utilizamos RESTORE VERIFYONLY en un proceso separado para confirmar que el archivo es legible físicamente. Además, al consultar msdb.dbo.backupset, garantizamos que siempre estamos verificando el archivo más reciente sin necesidad de modificar el código manualmente, y el registro en la tabla de auditoría proporciona una trazabilidad histórica del estado de salud de nuestras copias de seguridad.



Script:

### Escenarios A y B: Restauración por Desastre (Corrupción o Pérdida de Archivos)

**Situación:** El archivo de datos (.mdf) se corrompió o alguien borró el archivo de log (.ldf). La base de datos no arranca. **Solución:** Debemos "chancar" (sobrescribir) la base de datos dañada con nuestra copia de seguridad buena.

USE master;

GO

PRINT '--- Iniciando Restauración por Desastre (Escenarios A y B) ---';

-- 1. Poner la base de datos en modo SINGLE\_USER

-- Esto desconecta a todos y permite sobrescribir la BD aunque esté "trabada" o dañada.

ALTER DATABASE [QhatuPERU] SET SINGLE\_USER WITH ROLLBACK IMMEDIATE;

-- 2. Restaurar usando WITH REPLACE

-- 'REPLACE' es la orden clave: le dice a SQL "Ignora que la BD ya existe, sobrescríbela".

RESTORE DATABASE [QhatuPERU]

FROM DISK = 'C:\practicas\bds\full\QhatuPERU\_FULL\_Para\_EscenarioB.bak' -- <--- Usa tu ruta real aquí

WITH

REPLACE,

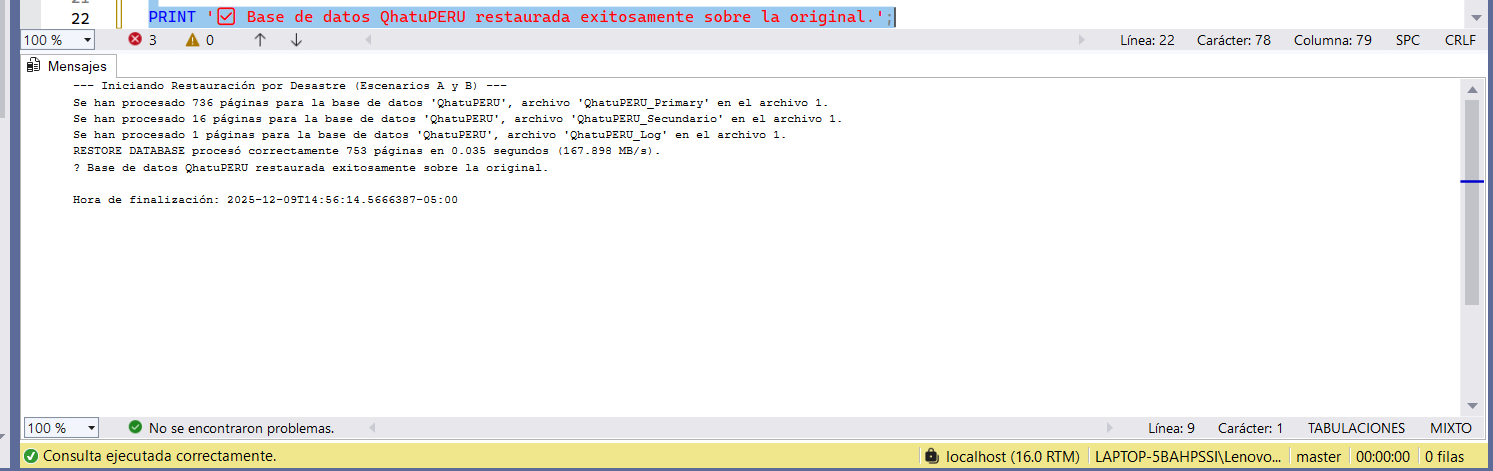
RECOVERY; -- Dejar la BD lista para usar

-- 3. Volver a modo MULTI\_USER para que todos entren

ALTER DATABASE [QhatuPERU] SET MULTI\_USER;

GO

PRINT '✅ Base de datos QhatuPERU restaurada exitosamente sobre la original.';



### Escenario C: Migración (Restauración en "Otro Servidor")

**Situación:** Necesitas llevar la base de datos a otro servidor (o crear una copia en el mismo servidor con otro nombre) y cambiar la ubicación de los archivos. **Solución:** Usar la opción WITH MOVE para reubicar los archivos físicos.

*Nota: Simularemos la migración creando una copia llamada QhatuPERU\_Migrada en una carpeta nueva.*

USE master;

GO

PRINT '--- Iniciando Migración Simulada (Escenario C) - CORREGIDO ---';

RESTORE DATABASE [QhatuPERU\_Migrada]

FROM DISK = 'C:\practicas\bds\full\QhatuPERU\_FULL\_Para\_EscenarioB.bak'

WITH

RECOVERY,

-- Mover el archivo de Datos Principal (.mdf)

MOVE 'QhatuPERU\_Primary' TO 'C:\practicas\bds\migracion\QhatuPERU\_Migrada.mdf',

-- Mover el archivo de Datos Secundario (.ndf) - ¡Este faltaba!

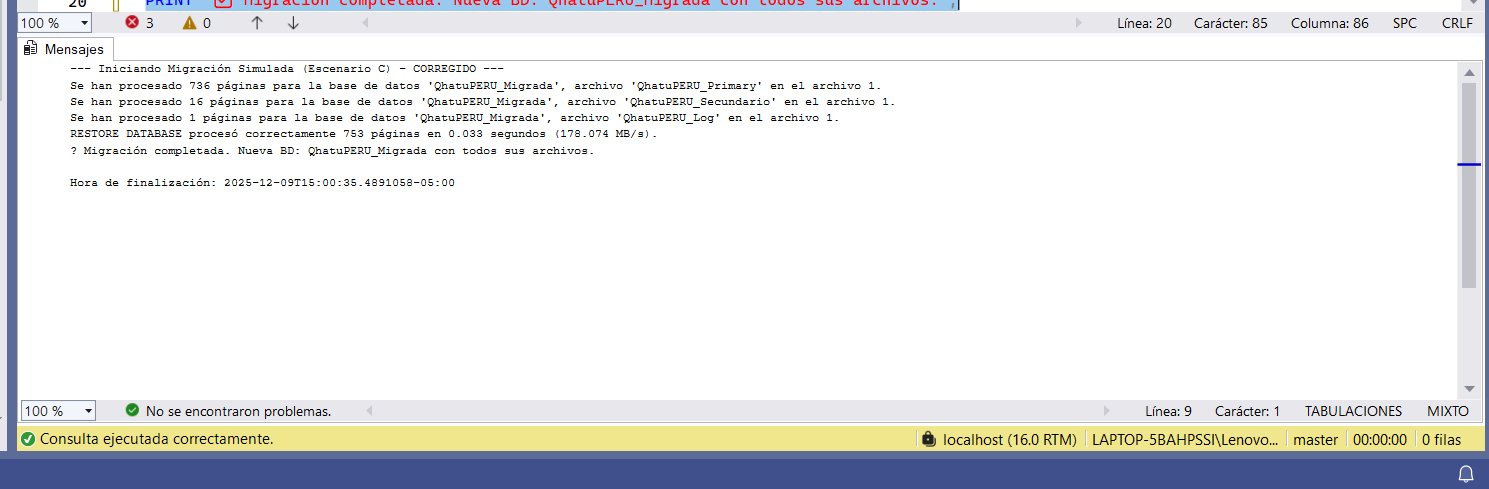
MOVE 'QhatuPERU\_Secundario' TO 'C:\practicas\bds\migracion\QhatuPERU\_Migrada\_Sec.ndf',

-- Mover el archivo de Log (.ldf)

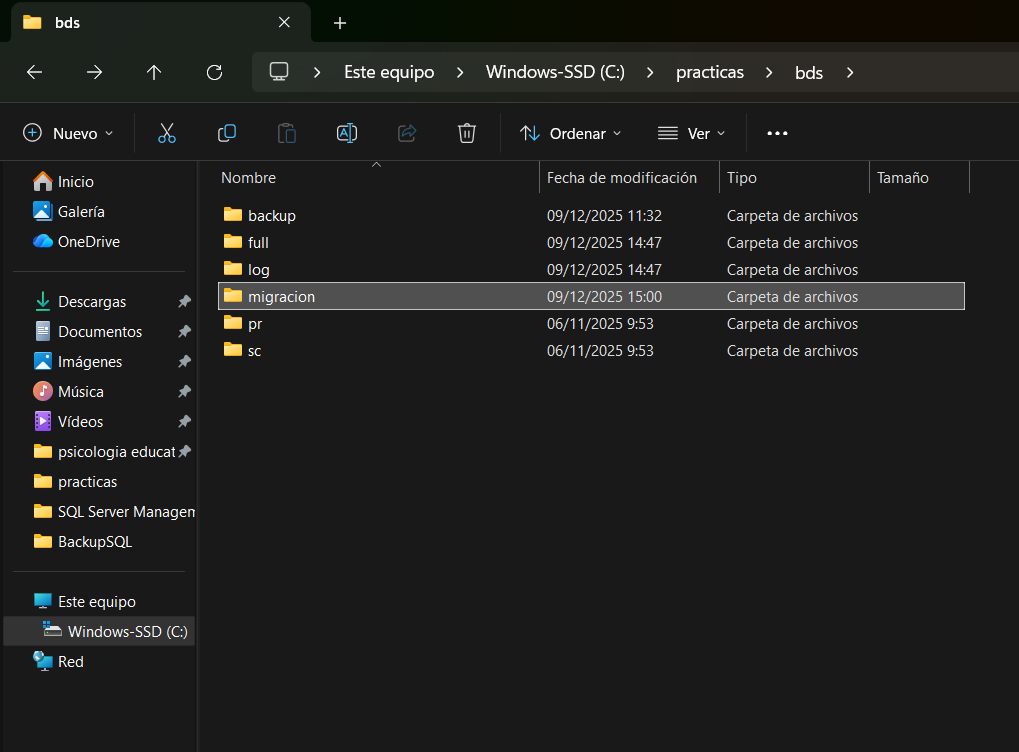
MOVE 'QhatuPERU\_Log' TO 'C:\practicas\bds\migracion\QhatuPERU\_Migrada\_log.ldf';

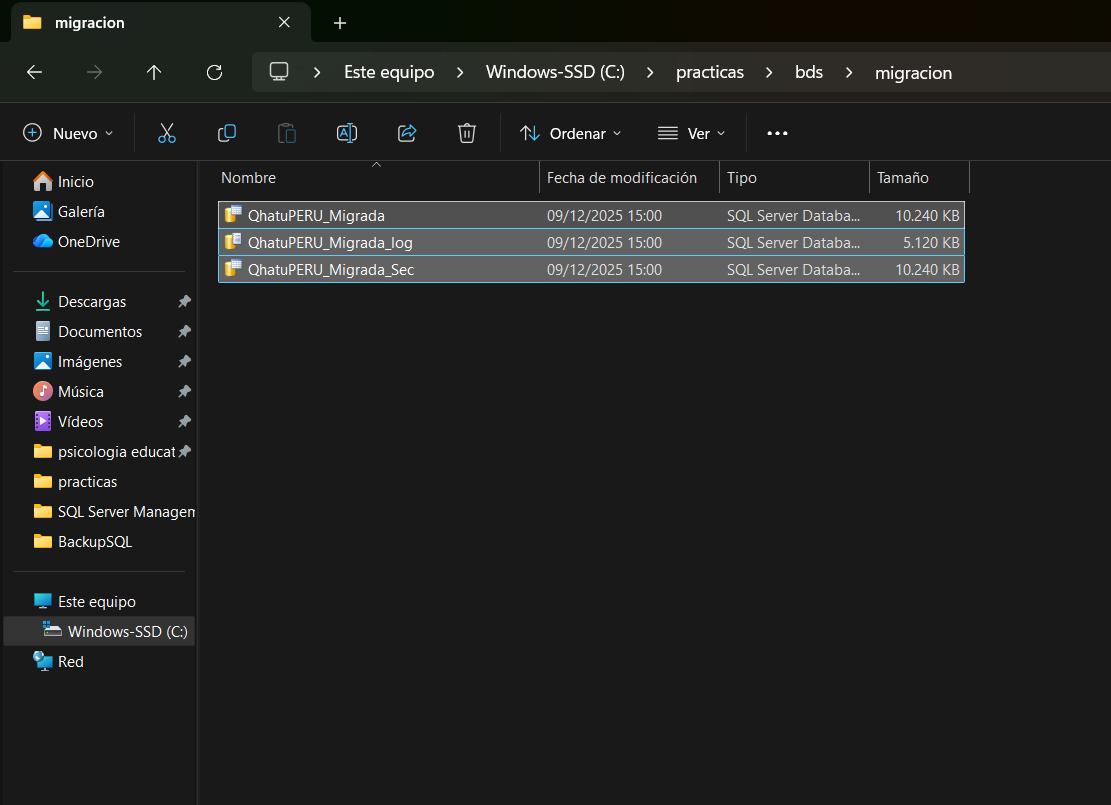
GO

PRINT '✅ Migración completada. Nueva BD: QhatuPERU\_Migrada con todos sus archivos.';



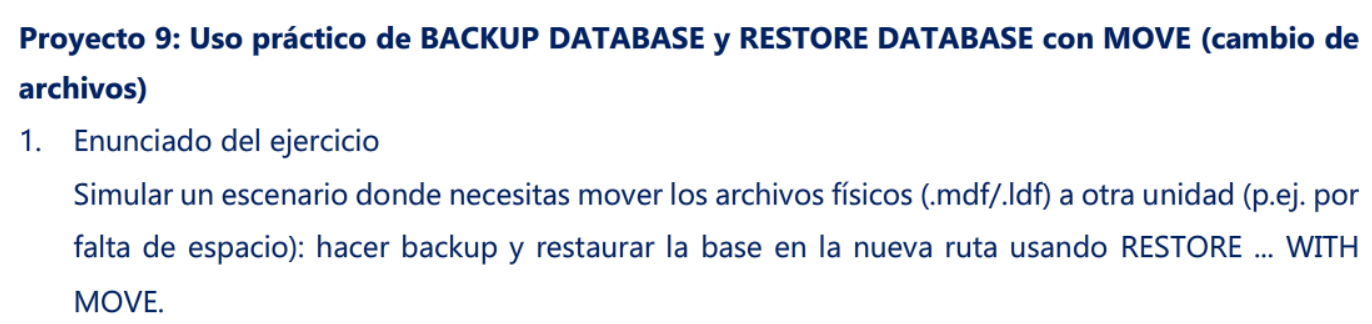
Justificación:





Buenas Prácticas:

En procedimientos de recuperación y migración, el uso de WITH REPLACE es la práctica estándar para recuperaciones destructivas donde la prioridad es restablecer el servicio sobre una base de datos corrupta existente. Por otro lado, WITH MOVE es la cláusula fundamental para migraciones, ya que permite restaurar una base de datos en un entorno con estructura de directorios diferente (como otro servidor) sin causar conflictos de rutas. Finalmente, siempre se debe consultar RESTORE FILELISTONLY antes de una migración para conocer los nombres lógicos exactos de los archivos y evitar errores de mapeo.



Script:

Este script hace dos cosas:

1. Genera un backup fresco (para asegurarnos de tener la última versión).
2. Restaura ese backup creando una **nueva base de datos** (QhatuPERU\_Moved) cuyos archivos físicos vivirán en la carpeta nueva.

USE master;

GO

PRINT '--- Proyecto 9: Mover Archivos por Falta de Espacio ---';

-- 1. Generar un Backup Rápido (El "paquete" que vamos a mover)

-- Si ya tienes uno reciente, puedes saltar este paso, pero mejor prevenir.

BACKUP DATABASE [QhatuPERU]

TO DISK = 'C:\practicas\bds\full\QhatuPERU\_Para\_Mover.bak'

WITH FORMAT, NAME = 'Backup para Mudanza';

GO

-- 2. Restaurar en la NUEVA UBICACIÓN (La "Mudanza")

-- Creamos una base llamada 'QhatuPERU\_Moved' para no chancar tu original.

RESTORE DATABASE [QhatuPERU\_Moved]

FROM DISK = 'C:\practicas\bds\full\QhatuPERU\_Para\_Mover.bak'

WITH

RECOVERY,

-- Aquí ocurre la magia: Redirigimos cada archivo a la carpeta vacía

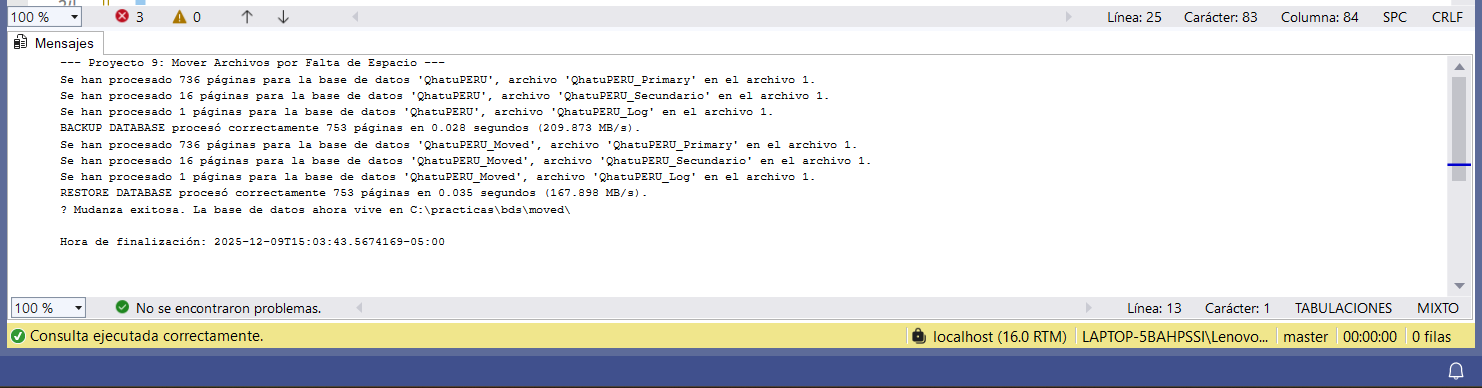
MOVE 'QhatuPERU\_Primary' TO 'C:\practicas\bds\moved\QhatuPERU\_Moved.mdf',

MOVE 'QhatuPERU\_Secundario' TO 'C:\practicas\bds\moved\QhatuPERU\_Moved\_Sec.ndf',

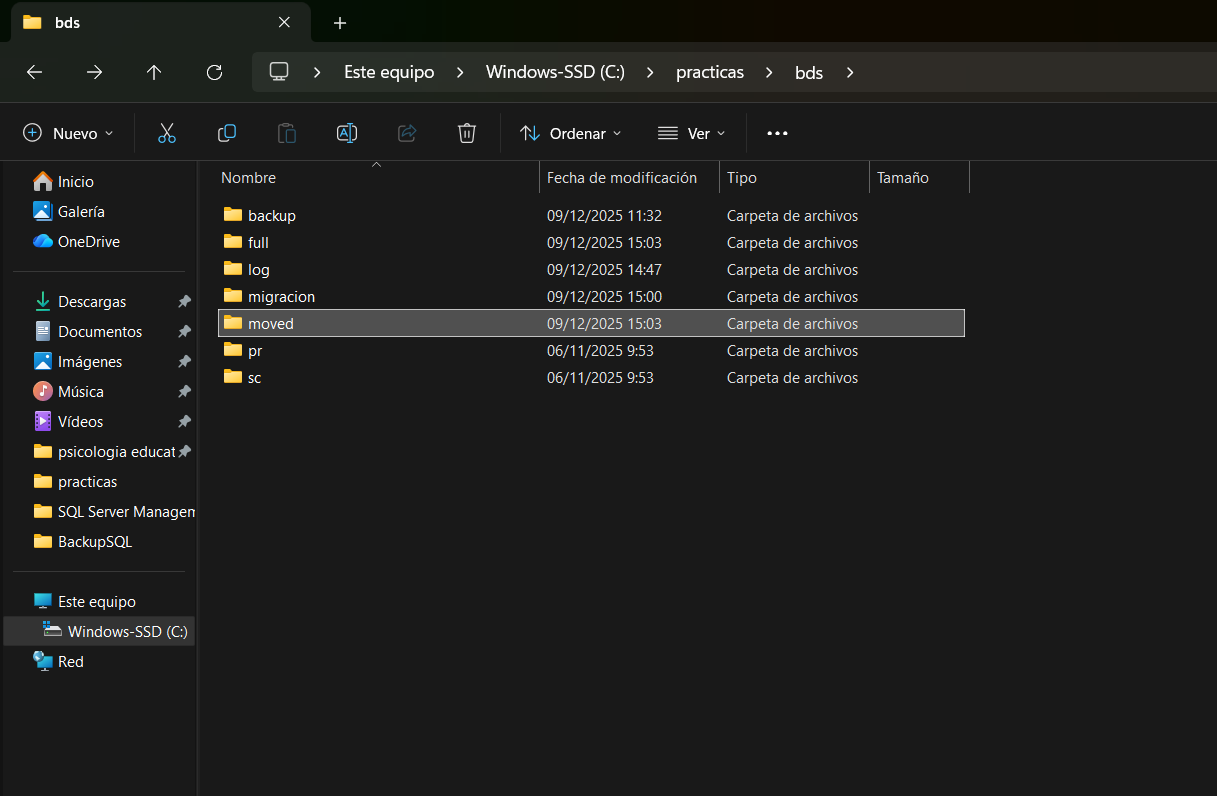
MOVE 'QhatuPERU\_Log' TO 'C:\practicas\bds\moved\QhatuPERU\_Moved\_log.ldf';

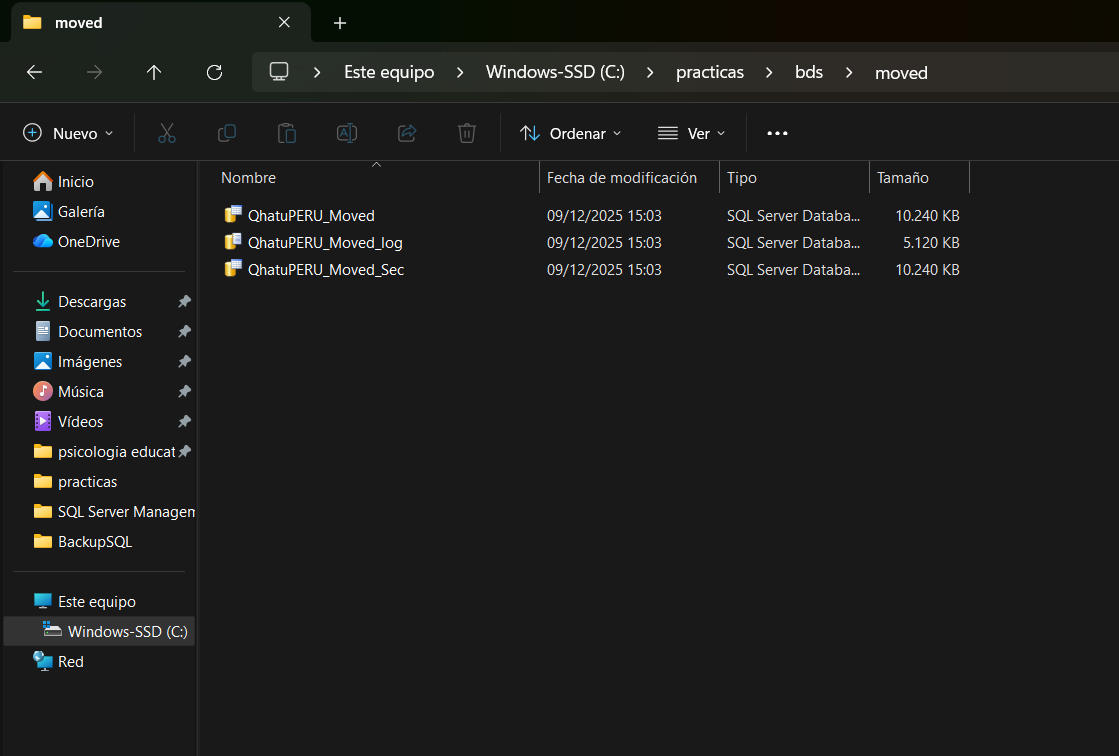
GO

PRINT '✅ Mudanza exitosa. La base de datos ahora vive en C:\practicas\bds\moved\';



Justificación:

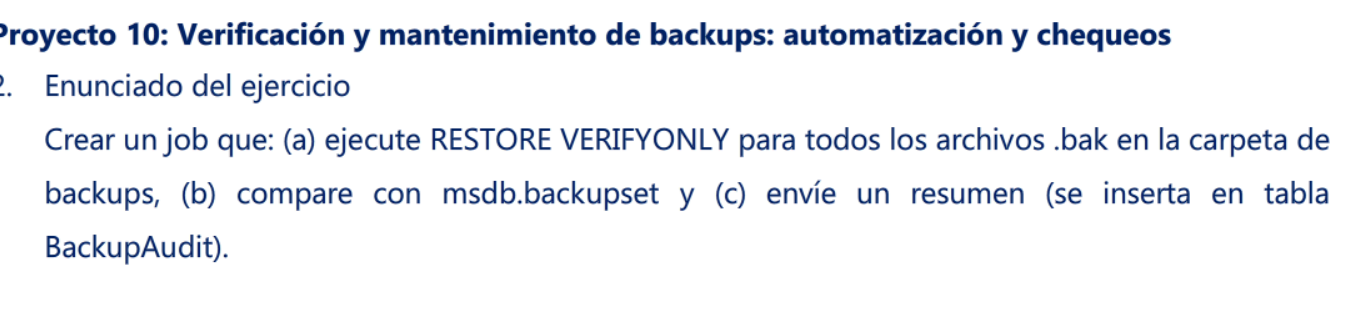






Buenas Prácticas:

La práctica fundamental aquí es el uso de RESTORE FILELISTONLY (implícito al conocer los nombres lógicos) para identificar los componentes de la base de datos y el uso preciso de WITH MOVE para la gestión del almacenamiento. En entornos de producción, esta técnica es vital cuando una unidad de disco alcanza su capacidad máxima; permite al DBA restaurar el servicio moviendo los archivos de datos (.mdf) o de log (.ldf) a unidades con mayor espacio libre sin perder la configuración ni los datos históricos.



Script:

Este script crea una tabla de auditoría (si no existe) y luego utiliza un cursor para recorrer el historial de backups de los últimos 7 días registrados en msdb, verificando la integridad de cada archivo físico.

USE [QhatuPERU];

GO

-- 1. Crear tabla de Auditoría (BackupAudit) si no existe

IF OBJECT\_ID('dbo.BackupAudit') IS NULL

BEGIN

CREATE TABLE dbo.BackupAudit (

ID INT IDENTITY(1,1) PRIMARY KEY,

FechaVerificacion DATETIME DEFAULT GETDATE(),

NombreArchivo NVARCHAR(500),

Estado VARCHAR(20), -- 'VALIDO' o 'CORRUPTO'

Mensaje NVARCHAR(MAX)

);

END

GO

PRINT '--- Iniciando Verificación Masiva de Backups (Últimos 7 días) ---';

DECLARE @RutaArchivo NVARCHAR(500);

DECLARE @MensajeError NVARCHAR(MAX);

DECLARE @EstadoFinal VARCHAR(20);

-- 2. Cursor para obtener todos los backups de QhatuPERU de la última semana

DECLARE InspectorBackups CURSOR LOCAL FOR

SELECT DISTINCT bmf.physical\_device\_name

FROM msdb.dbo.backupset bs

INNER JOIN msdb.dbo.backupmediafamily bmf ON bs.media\_set\_id = bmf.media\_set\_id

WHERE bs.database\_name = 'QhatuPERU'

AND bs.backup\_finish\_date >= DATEADD(day, -7, GETDATE()) -- Solo revisar lo reciente

AND bmf.physical\_device\_name NOT LIKE 'NULL'; -- Evitar errores

OPEN InspectorBackups;

FETCH NEXT FROM InspectorBackups INTO @RutaArchivo;

WHILE @@FETCH\_STATUS = 0

BEGIN

PRINT 'Chequeando: ' + @RutaArchivo;

-- Reiniciar variables

SET @MensajeError = 'Verificación Exitosa';

SET @EstadoFinal = 'VALIDO';

BEGIN TRY

-- 3. El comando clave: Verificar integridad sin restaurar

RESTORE VERIFYONLY FROM DISK = @RutaArchivo;

END TRY

BEGIN CATCH

-- Si falla, capturamos el error

SET @EstadoFinal = 'CORRUPTO';

SET @MensajeError = ERROR\_MESSAGE();

PRINT '❌ ERROR detectado en: ' + @RutaArchivo;

END CATCH

-- 4. Insertar resultado en la tabla BackupAudit

INSERT INTO dbo.BackupAudit (NombreArchivo, Estado, Mensaje)

VALUES (@RutaArchivo, @EstadoFinal, @MensajeError);

FETCH NEXT FROM InspectorBackups INTO @RutaArchivo;

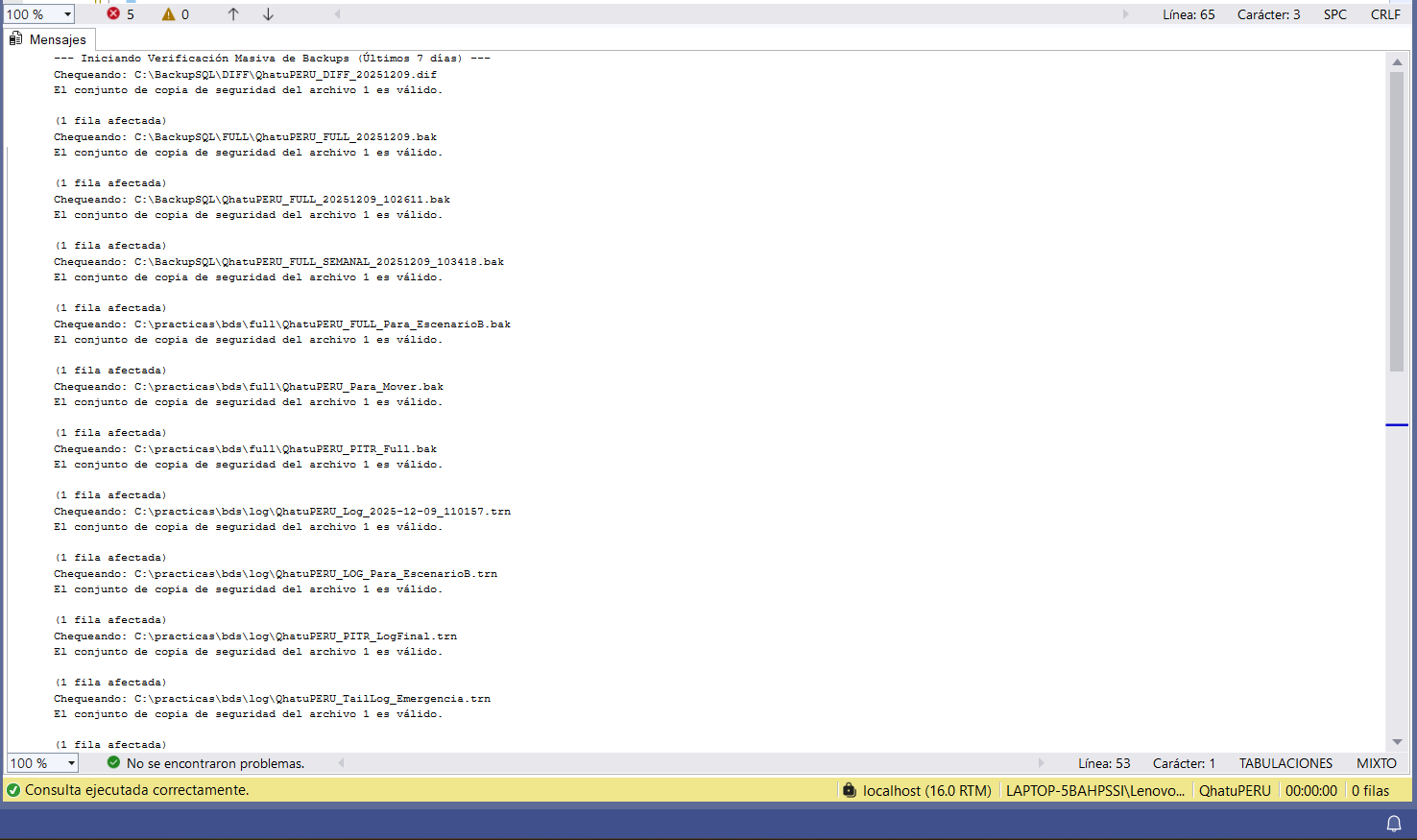
END

CLOSE InspectorBackups;

DEALLOCATE InspectorBackups;

PRINT '--- Auditoría Finalizada. Revisa la tabla BackupAudit. ---';

GO

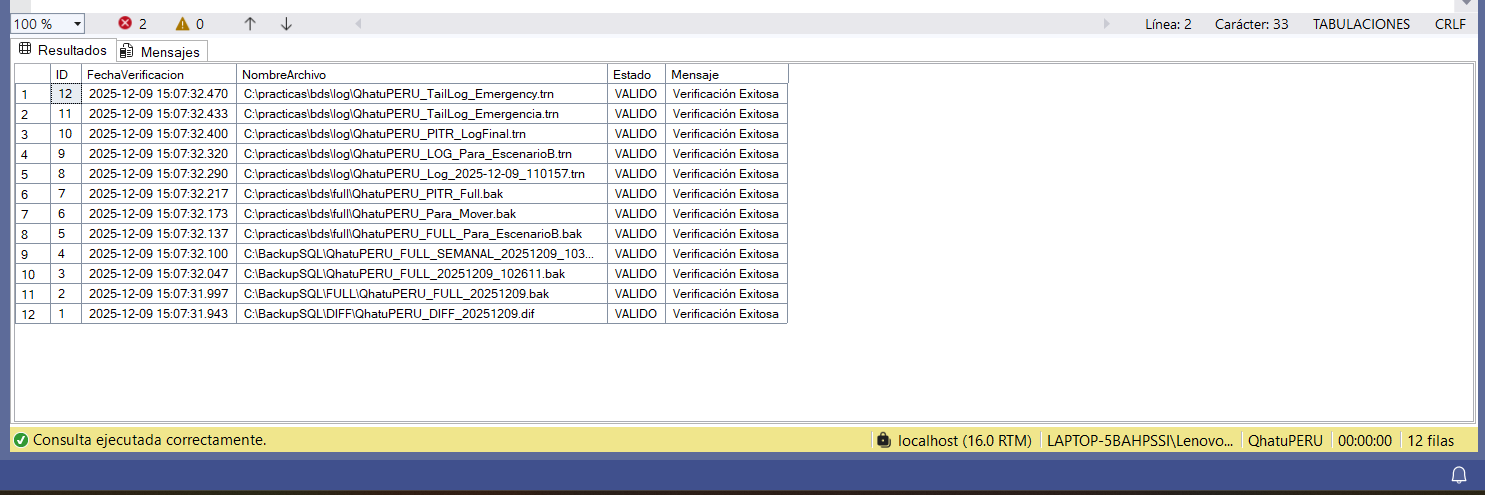




Justificación:

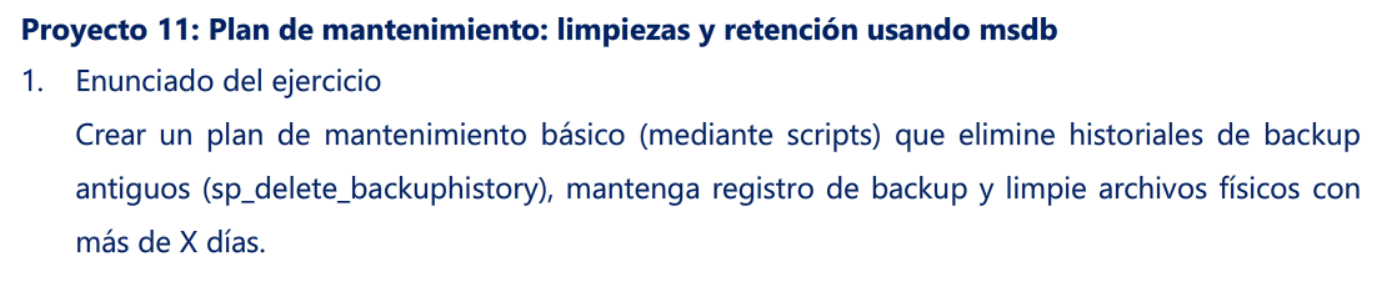
SELECT \* FROM [QhatuPERU].dbo.BackupAudit

ORDER BY FechaVerificacion DESC;



Buenas Prácticas:

Este procedimiento aplica la Validación en Lote (Bulk Verification) y la Trazabilidad. En lugar de confiar ciegamente en que "si el backup se hizo, está bien", este proceso re-valida periódicamente los archivos almacenados (RESTORE VERIFYONLY), asegurando que no hayan sufrido "bit rot" (deterioro de datos) o corrupción de disco con el tiempo. Al cruzar la información con msdb (la base de datos del sistema), garantizamos que estamos auditando los archivos que SQL Server reconoce oficialmente como parte de su historial de recuperación.



Script:

Este script está diseñado para ejecutarse periódicamente (por ejemplo, una vez a la semana). Define una variable de retención (ej. 30 días) y elimina todo lo que sea más viejo que eso.

USE master;

GO

-- Configuración: Define cuántos días de historia quieres guardar

DECLARE @DiasRetencion INT = 30; -- Guardar últimos 30 días

DECLARE @FechaLimite DATETIME;

DECLARE @RutaArchivo NVARCHAR(512);

DECLARE @ComandoDel NVARCHAR(1000);

-- Calcular la fecha de corte

SET @FechaLimite = DATEADD(dd, -@DiasRetencion, GETDATE());

PRINT '=== INICIANDO PLAN DE MANTENIMIENTO: LIMPIEZA Y RETENCIÓN ===';

PRINT 'Fecha de corte: ' + CONVERT(NVARCHAR, @FechaLimite, 120);

-- PARTE 1: Limpieza de Archivos Físicos (Disco)

-- Usamos el cursor con el JOIN correcto que ya sabemos que funciona

DECLARE LimpiezaCursor CURSOR LOCAL FOR

SELECT

bmf.physical\_device\_name

FROM msdb.dbo.backupset AS bs

INNER JOIN msdb.dbo.backupmediafamily AS bmf

ON bs.media\_set\_id = bmf.media\_set\_id

WHERE

bs.database\_name = 'QhatuPERU'

AND bs.backup\_finish\_date < @FechaLimite

AND bmf.physical\_device\_name NOT LIKE 'NULL'; -- Evitar errores

OPEN LimpiezaCursor;

FETCH NEXT FROM LimpiezaCursor INTO @RutaArchivo;

WHILE @@FETCH\_STATUS = 0

BEGIN

-- Intentar borrar el archivo físico

SET @ComandoDel = N'DEL /F /Q "' + @RutaArchivo + N'"';

PRINT 'Eliminando archivo físico: ' + @RutaArchivo;

-- Ejecutar comando de sistema (xp\_cmdshell debe estar activo)

BEGIN TRY

EXEC master..xp\_cmdshell @ComandoDel, NO\_OUTPUT;

END TRY

BEGIN CATCH

PRINT ' - No se pudo borrar el archivo (quizás ya no existe).';

END CATCH

FETCH NEXT FROM LimpiezaCursor INTO @RutaArchivo;

END

CLOSE LimpiezaCursor;

DEALLOCATE LimpiezaCursor;

PRINT '✅ Limpieza física completada.';

-- PARTE 2: Limpieza de Historial (Metadatos en MSDB)

-- Esto borra los registros en las tablas internas de SQL Server

PRINT 'Limpiando historial en MSDB...';

EXEC msdb.dbo.sp\_delete\_backuphistory @FechaLimite;

PRINT '✅ Historial antiguo eliminado correctamente.';

PRINT '=== PLAN DE MANTENIMIENTO FINALIZADO ===';

GO



Justificación:

SELECT MIN(backup\_finish\_date) AS [Fecha del Backup Más Antiguo en Historial]

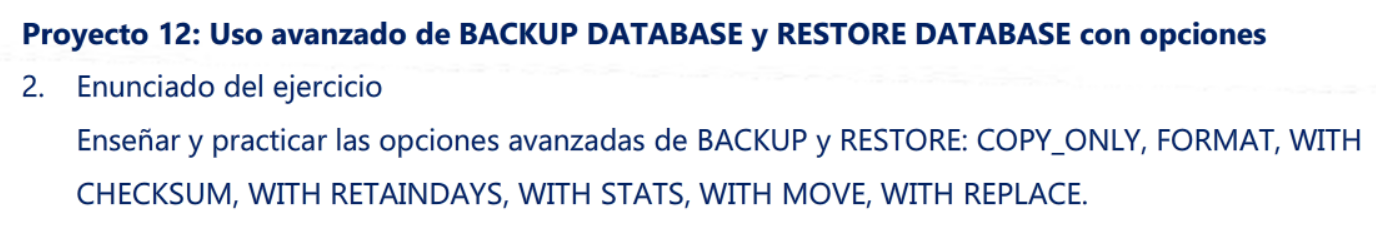
FROM msdb.dbo.backupset

WHERE database\_name = 'QhatuPERU';



Buenas Prácticas:

Este plan de mantenimiento implementa una Política de Retención (Retention Policy) clara, que es fundamental para evitar el llenado descontrolado del disco duro y el crecimiento excesivo de la base de datos del sistema msdb. Al combinar la limpieza física (xp\_cmdshell + cursor) con la limpieza lógica (sp\_delete\_backuphistory) en un solo proceso, se garantiza la coherencia entre lo que existe en el disco y lo que SQL Server registra, manteniendo el entorno optimizado y libre de "basura" digital obsoleta.



Script:

**WITH CHECKSUM y STATS (La Seguridad Primero)**

**Para qué sirve:** CHECKSUM verifica que la página de datos no esté dañada *antes* de guardarla en el backup. Es tu seguro de vida contra datos corruptos. STATS te muestra el progreso.

USE master;

GO

PRINT '--- 1. Backup con Verificación de Checksum ---';

BACKUP DATABASE [QhatuPERU]

TO DISK = 'C:\practicas\bds\full\QhatuPERU\_Seguro.bak'

WITH

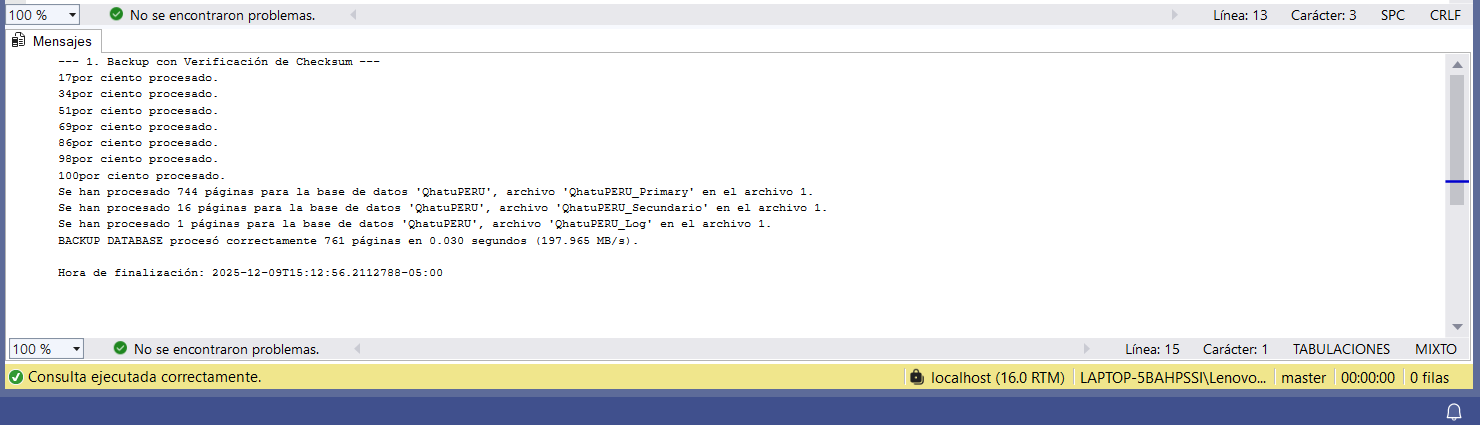
FORMAT, -- Sobrescribe cualquier backup previo en este archivo

CHECKSUM, -- Valida integridad de datos AL LEER

STATS = 10, -- Muestra progreso cada 10%

NAME = 'Backup Verificado con Checksum';

GO



**COPY\_ONLY (El Backup "Fantasma")**

**Para qué sirve:** Haces este backup cuando te piden "una copia rápida para desarrollo", pero **NO quieres romper tu cadena de logs** (Liferencial/Log). Este backup no afecta la secuencia de restauraciones.

PRINT '--- 2. Backup Copy-Only (Sin romper la cadena) ---';

BACKUP DATABASE [QhatuPERU]

TO DISK = 'C:\practicas\bds\full\QhatuPERU\_CopyOnly.bak'

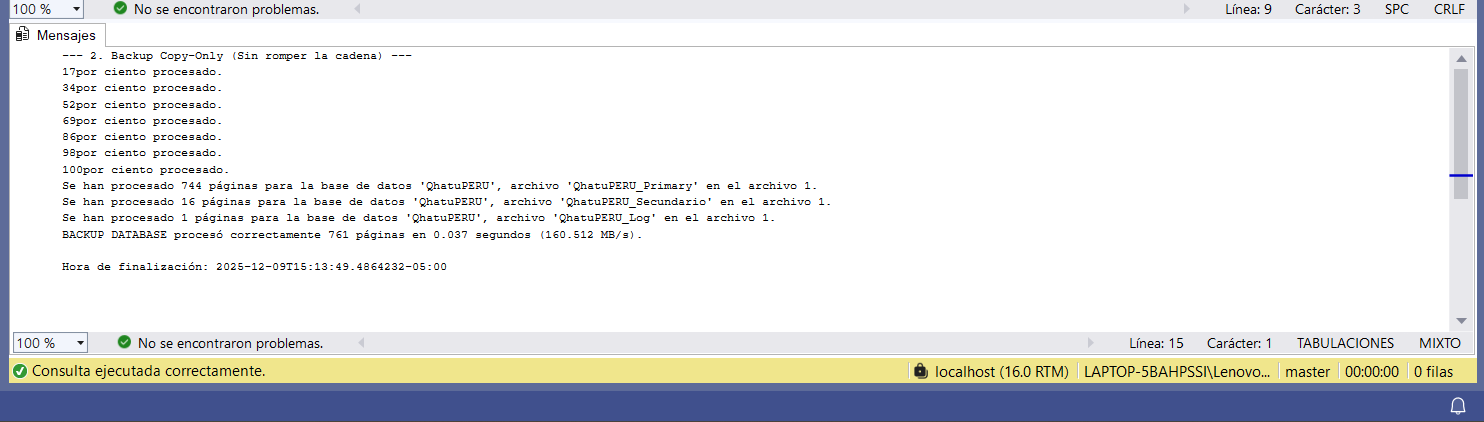
WITH

COPY\_ONLY, -- ¡La clave! No afecta los backups diferenciales futuros

STATS = 10,

NAME = 'Copia para Desarrolladores';

GO



**WITH RETAINDAYS (Protección contra Borrado Accidental)**

**Para qué sirve:** Le pones un "candado" al archivo. SQL Server te impedirá sobrescribirlo hasta que pasen los días que definas.

PRINT '--- 3. Backup con Retención (Protegido 7 días) ---';

BACKUP DATABASE [QhatuPERU]

TO DISK = 'C:\practicas\bds\full\QhatuPERU\_Protegido.bak'

WITH

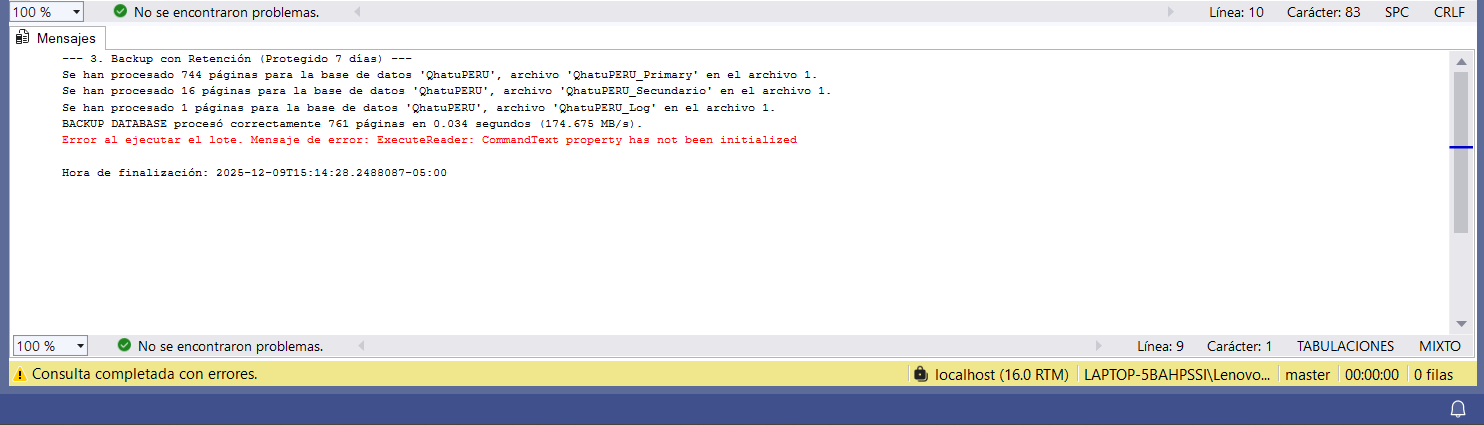
FORMAT,

RETAINDAYS = 7, -- No se podrá sobrescribir este archivo hasta dentro de 7 días

NAME = 'Backup Intocable';

GO

-- Si intentas ejecutar esto de nuevo inmediatemente, fallará protegiendo tu data.





**Restauración Avanzada: MOVE + REPLACE**

**Para qué sirve:** Ya los has usado, pero aquí combinamos todo para forzar una restauración destructiva sobre una base de datos existente, moviendo archivos al mismo tiempo.

PRINT '--- 4. Restauración Agresiva (Replace + Move) ---';

-- Simulamos que restauramos sobre la base "QhatuPERU\_Moved" que creamos antes

RESTORE DATABASE [QhatuPERU\_Moved]

FROM DISK = 'C:\practicas\bds\full\QhatuPERU\_Seguro.bak'

WITH

RECOVERY,

REPLACE, -- "No me importa que ya exista, cháncala"

STATS = 20,

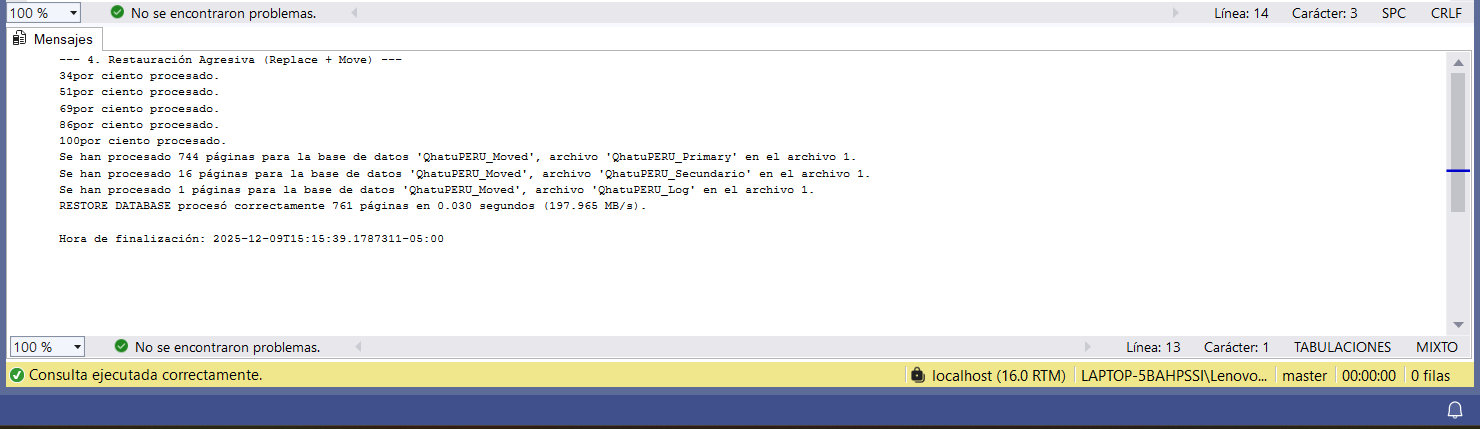
-- Recuerda usar los nombres lógicos correctos que descubrimos en ejercicios pasados

MOVE 'QhatuPERU\_Primary' TO 'C:\practicas\bds\moved\QhatuPERU\_Moved.mdf',

MOVE 'QhatuPERU\_Secundario' TO 'C:\practicas\bds\moved\QhatuPERU\_Moved\_Sec.ndf',

MOVE 'QhatuPERU\_Log' TO 'C:\practicas\bds\moved\QhatuPERU\_Moved\_log.ldf';

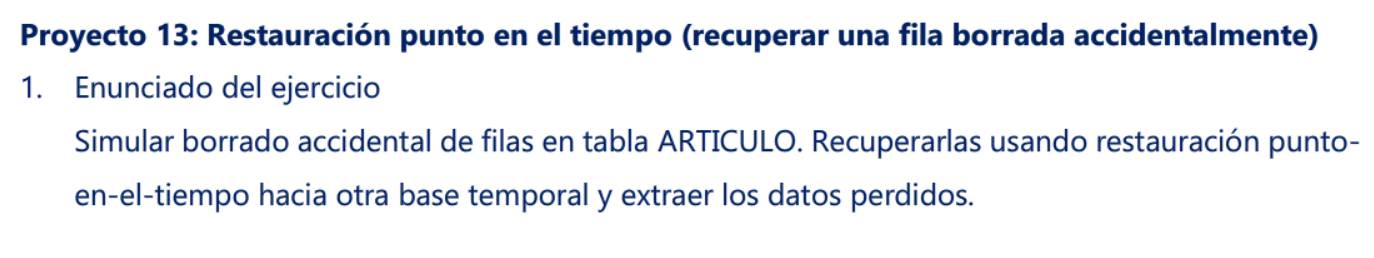
GO



Buenas Prácticas:

En este nivel avanzado, cada opción tiene un propósito estratégico de administración:

* **CHECKSUM**: Úsalo **siempre** en producción. Es mejor que el backup falle y te avise a tiempo, a que guardes datos corruptos y te des cuenta meses después al intentar restaurar.
* **COPY\_ONLY**: Úsalo para backups ad-hoc (fuera de programa). Evita que el siguiente Backup Diferencial sea gigante o que la cadena de logs se rompa.
* **RETAINDAYS**: Es vital para cintas o almacenamiento a largo plazo, evitando que scripts automáticos borren cierres de mes o año importantes.



Script:

**Simulación del Desastre (Setup)**

Ejecuta este script para crear el escenario: tabla, datos, backup y el borrado accidental.

USE [QhatuPERU];

GO

PRINT '--- 🧹 INICIANDO LIMPIEZA DE DEPENDENCIAS ---';

-- 1. ROMPER VÍNCULOS (Limpieza de Foreign Keys)

-- Este código busca cualquier tabla que apunte a 'ARTICULO' y elimina la restricción

DECLARE @sql NVARCHAR(MAX) = N'';

SELECT @sql += N'ALTER TABLE ' + QUOTENAME(OBJECT\_SCHEMA\_NAME(parent\_object\_id))

+ N'.' + QUOTENAME(OBJECT\_NAME(parent\_object\_id)) +

N' DROP CONSTRAINT ' + QUOTENAME(name) + N'; '

FROM sys.foreign\_keys

WHERE referenced\_object\_id = OBJECT\_ID(N'dbo.ARTICULO');

-- Si encontró algo, lo ejecuta

EXEC sp\_executesql @sql;

PRINT '✅ Restricciones Foreign Key eliminadas (si existían).';

-- 2. AHORA SÍ: BORRAR LA TABLA VIEJA

IF OBJECT\_ID('dbo.ARTICULO') IS NOT NULL

DROP TABLE dbo.ARTICULO;

PRINT '✅ Tabla ARTICULO antigua eliminada.';

GO

PRINT '--- 🚀 INICIANDO ESCENARIO DEL EJERCICIO ---';

-- 3. CREAR LA TABLA CORRECTA

CREATE TABLE dbo.ARTICULO (

ID INT PRIMARY KEY,

Producto VARCHAR(100),

Precio DECIMAL(10,2)

);

-- 4. INSERTAR DATOS

INSERT INTO dbo.ARTICULO (ID, Producto, Precio)

VALUES

(1, 'Laptop Gamer', 1500.00),

(2, 'Mouse Inalámbrico', 25.00),

(3, 'Teclado Mecánico', 80.00);

PRINT '✅ Tabla creada y datos insertados.';

-- 5. BACKUP FULL BASE

BACKUP DATABASE [QhatuPERU]

TO DISK = 'C:\practicas\bds\full\QhatuPERU\_P13\_Full.bak'

WITH FORMAT, NAME = 'Full Backup P13';

GO

-- Espera técnica para diferenciar tiempos

WAITFOR DELAY '00:00:02';

-- 6. CAPTURAR LA HORA SEGURA (¡COPIA ESTO!)

DECLARE @HoraSegura DATETIME = GETDATE();

PRINT '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*';

PRINT 'COPIA ESTA HORA EXACTA PARA EL SIGUIENTE PASO:';

PRINT CONVERT(VARCHAR(30), @HoraSegura, 126);

PRINT '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*';

-- Espera técnica

WAITFOR DELAY '00:00:02';

-- 7. EL ACCIDENTE (Borrar ID 2)

DELETE FROM dbo.ARTICULO WHERE ID = 2;

PRINT '❌ Desastre simulado: Se borró el Mouse.';

-- 8. BACKUP DEL LOG (Captura el desastre)

BACKUP LOG [QhatuPERU]

TO DISK = 'C:\practicas\bds\log\QhatuPERU\_P13\_Log.trn'

WITH FORMAT, NAME = 'Log Post-Desastre';

GO



**Restauración a una Base Temporal (Side-by-Side)**

Ahora restauraremos la base de datos con otro nombre (QhatuPERU\_Temp) deteniéndonos en la hora que copiaste.

USE master;

GO

DECLARE @HoraRestauracion DATETIME;

SET @HoraRestauracion = '2025-12-09T15:23:26.017'; -- Ej: '2025-12-09T15:00:00.123'

PRINT '--- Restaurando QhatuPERU\_Temp hasta antes del borrado ---';

-- 1. Restaurar FULL en una NUEVA base de datos (con MOVE para no chocar archivos)

RESTORE DATABASE [QhatuPERU\_Temp]

FROM DISK = 'C:\practicas\bds\full\QhatuPERU\_P13\_Full.bak'

WITH

NORECOVERY,

MOVE 'QhatuPERU\_Primary' TO 'C:\practicas\bds\moved\QhatuPERU\_Temp.mdf',

MOVE 'QhatuPERU\_Secundario' TO 'C:\practicas\bds\moved\QhatuPERU\_Temp\_Sec.ndf',

MOVE 'QhatuPERU\_Log' TO 'C:\practicas\bds\moved\QhatuPERU\_Temp\_Log.ldf';

-- 2. Restaurar LOG con STOPAT (Viajamos al pasado)

RESTORE LOG [QhatuPERU\_Temp]

FROM DISK = 'C:\practicas\bds\log\QhatuPERU\_P13\_Log.trn'

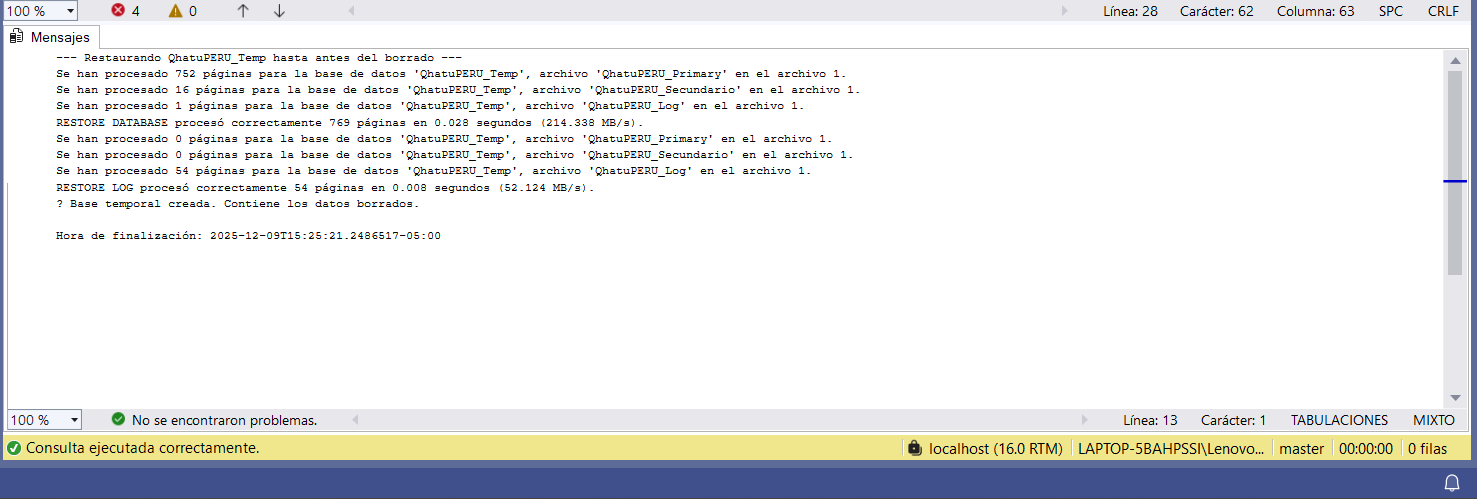
WITH

STOPAT = @HoraRestauracion,

RECOVERY;

GO

PRINT '✅ Base temporal creada. Contiene los datos borrados.';



**Recuperación de Datos (El Rescate)**

Ahora tenemos dos bases: QhatuPERU (le falta el Mouse) y QhatuPERU\_Temp (tiene el Mouse). Haremos una consulta para insertar lo que falta.

USE [QhatuPERU];

GO

PRINT '--- Iniciando Rescate de Datos ---';

-- Insertamos en la tabla original lo que existe en la temporal pero NO en la original

INSERT INTO dbo.ARTICULO (ID, Producto, Precio)

SELECT ID, Producto, Precio

FROM [QhatuPERU\_Temp].dbo.ARTICULO

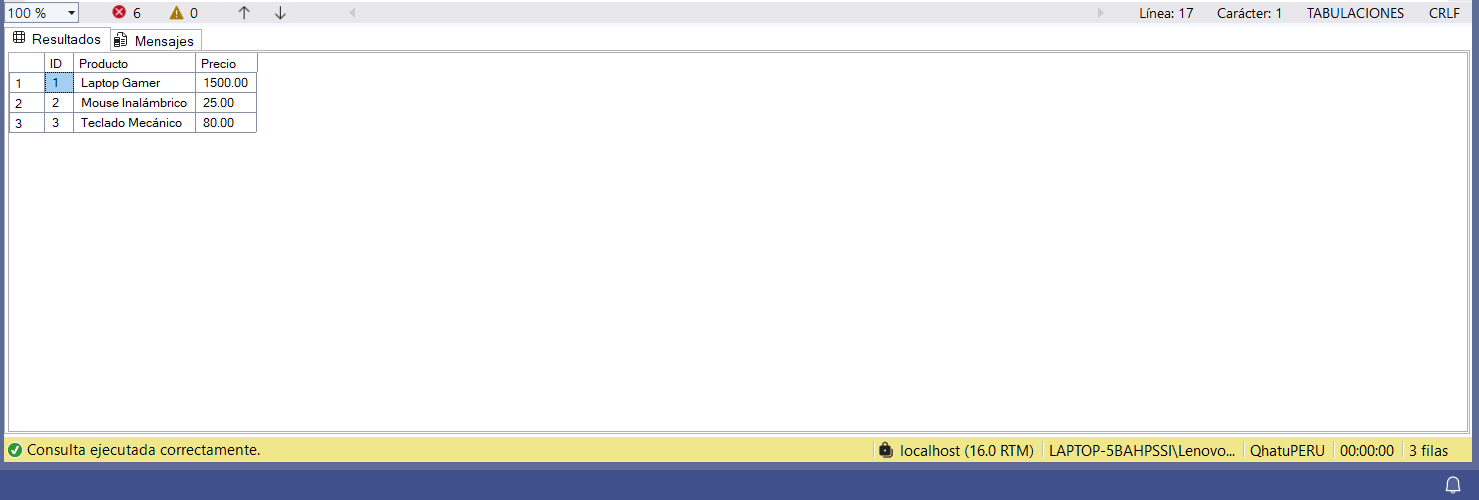
WHERE ID NOT IN (SELECT ID FROM [QhatuPERU].dbo.ARTICULO);

PRINT '✅ ¡Datos recuperados! El Mouse ha vuelto.';

-- Verificación final

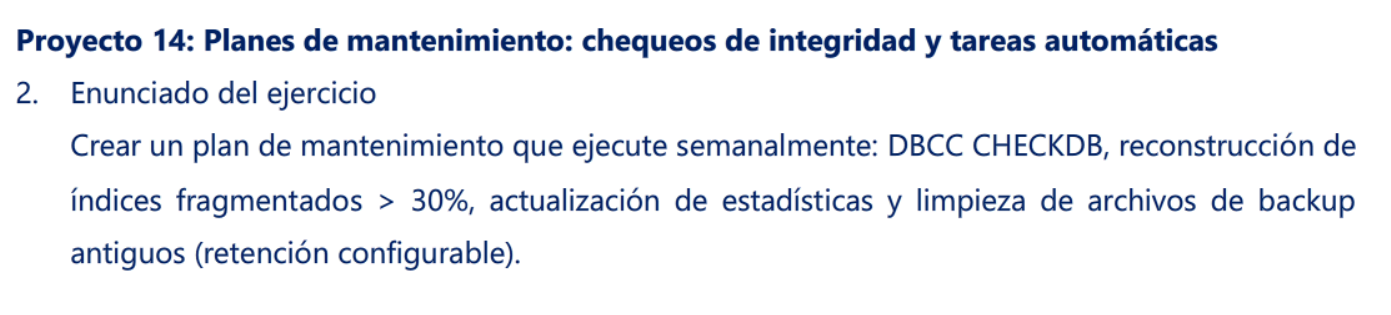
SELECT \* FROM dbo.ARTICULO;

GO



Buenas Prácticas:

Esta técnica se conoce como Restauración Side-by-Side (Lado a Lado). Es la práctica recomendada para recuperar datos específicos (como una tabla o filas borradas) sin afectar la disponibilidad de la base de datos principal ni perder las transacciones válidas ocurridas después del error. Al restaurar en una base temporal con STOPAT y usar INSERT INTO ... SELECT, realizamos una "cirugía de precisión" en lugar de sobrescribir toda la base de datos de producción.



Script:

**Plan de Mantenimiento Semanal**

Este script realiza las 4 tareas solicitadas en secuencia. He incluido una lógica inteligente para detectar y reconstruir **solo** los índices que realmente lo necesitan (fragmentación > 30%), cumpliendo con el requisito del ejercicio.

USE [QhatuPERU];

GO

PRINT '=== INICIO DEL PLAN DE MANTENIMIENTO SEMANAL: QhatuPERU ===';

-- 1. CHEQUEO DE INTEGRIDAD (El "Check-Up" Médico)

-- DBCC CHECKDB busca corrupción física o lógica en la base de datos.

PRINT '>> 1. Verificando integridad de la base de datos...';

DBCC CHECKDB(N'QhatuPERU') WITH NO\_INFOMSGS;

PRINT '✅ Integridad verificada correctamente.';

-- 2. RECONSTRUCCIÓN DE ÍNDICES (Solo los fragmentados > 30%)

PRINT '>> 2. Analizando y optimizando índices fragmentados...';

DECLARE @NombreTabla NVARCHAR(255);

DECLARE @NombreIndice NVARCHAR(255);

DECLARE @PorcFragmentacion FLOAT;

DECLARE @ComandoSQL NVARCHAR(MAX);

-- Cursor inteligente: Busca en las estadísticas internas qué índices están "desordenados" (fragmentados)

DECLARE CursorIndices CURSOR FOR

SELECT

t.name AS TableName,

i.name AS IndexName,

s.avg\_fragmentation\_in\_percent

FROM sys.dm\_db\_index\_physical\_stats(DB\_ID(), NULL, NULL, NULL, 'LIMITED') s

JOIN sys.tables t ON s.object\_id = t.object\_id

JOIN sys.indexes i ON s.object\_id = i.object\_id AND s.index\_id = i.index\_id

WHERE s.avg\_fragmentation\_in\_percent > 30 -- ¡Aquí está la condición del 30%!

AND i.name IS NOT NULL

ORDER BY s.avg\_fragmentation\_in\_percent DESC;

OPEN CursorIndices;

FETCH NEXT FROM CursorIndices INTO @NombreTabla, @NombreIndice, @PorcFragmentacion;

WHILE @@FETCH\_STATUS = 0

BEGIN

PRINT ' - Reconstruyendo índice: ' + @NombreIndice + ' en tabla ' + @NombreTabla +

' (Fragmentación: ' + CONVERT(VARCHAR, CAST(@PorcFragmentacion AS INT)) + '%)';

-- Generamos y ejecutamos el comando de reconstrucción dinámicamente

SET @ComandoSQL = 'ALTER INDEX [' + @NombreIndice + '] ON [' + @NombreTabla + '] REBUILD';

EXEC(@ComandoSQL);

FETCH NEXT FROM CursorIndices INTO @NombreTabla, @NombreIndice, @PorcFragmentacion;

END

CLOSE CursorIndices;

DEALLOCATE CursorIndices;

PRINT '✅ Optimización de índices finalizada.';

-- 3. ACTUALIZACIÓN DE ESTADÍSTICAS

-- Ayuda a SQL Server a "conocer" los datos para hacer consultas rápidas.

PRINT '>> 3. Actualizando estadísticas...';

EXEC sp\_updatestats;

PRINT '✅ Estadísticas actualizadas.';

-- 4. LIMPIEZA DE BACKUPS ANTIGUOS (Retención Configurable)

PRINT '>> 4. Limpiando backups antiguos...';

DECLARE @DiasRetencion INT = 28; -- Configuración: Mantener historial de 4 semanas

DECLARE @FechaLimite DATETIME = DATEADD(dd, -@DiasRetencion, GETDATE());

DECLARE @ArchivoBorrar NVARCHAR(500);

DECLARE @CmdBorrar NVARCHAR(500);

-- A. Limpieza de Historial en MSDB

EXEC msdb.dbo.sp\_delete\_backuphistory @FechaLimite;

-- B. Limpieza de Archivos Físicos (Usando la técnica vista en proyectos anteriores)

DECLARE CursorBorrado CURSOR LOCAL FOR

SELECT DISTINCT bmf.physical\_device\_name

FROM msdb.dbo.backupset bs

JOIN msdb.dbo.backupmediafamily bmf ON bs.media\_set\_id = bmf.media\_set\_id

WHERE bs.database\_name = 'QhatuPERU' AND bs.backup\_finish\_date < @FechaLimite;

OPEN CursorBorrado;

FETCH NEXT FROM CursorBorrado INTO @ArchivoBorrar;

WHILE @@FETCH\_STATUS = 0

BEGIN

-- Borrar archivo físico

SET @CmdBorrar = 'DEL /F /Q "' + @ArchivoBorrar + '"';

BEGIN TRY

EXEC master..xp\_cmdshell @CmdBorrar, NO\_OUTPUT;

PRINT ' - Archivo eliminado por antigüedad: ' + @ArchivoBorrar;

END TRY

BEGIN CATCH

-- Ignoramos errores si el archivo ya no existe

END CATCH

FETCH NEXT FROM CursorBorrado INTO @ArchivoBorrar;

END

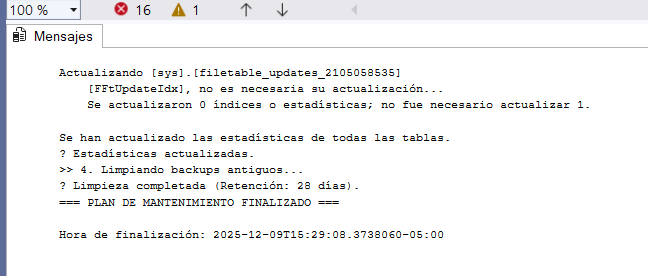
CLOSE CursorBorrado;

DEALLOCATE CursorBorrado;

PRINT '✅ Limpieza completada (Retención: ' + CAST(@DiasRetencion AS VARCHAR) + ' días).';

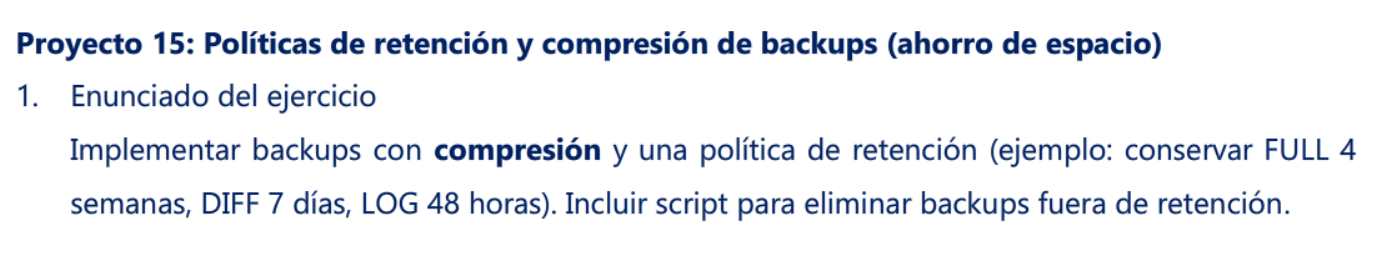
PRINT '=== PLAN DE MANTENIMIENTO FINALIZADO ===';

GO



Buenas Prácticas:

* **Integridad (CHECKDB)**: Se ejecuta primero para asegurar que no estemos optimizando una base de datos corrupta.
* **Mantenimiento Inteligente**: No reconstruimos *todos* los índices ciegamente, sino solo aquellos que superan el umbral del **30%** de fragmentación. Esto ahorra inmensos recursos de CPU y disco en comparación con un mantenimiento "tonto" que reconstruye todo siempre.
* **Estadísticas (sp\_updatestats)**: Se actualizan después de los índices para garantizar que el optimizador de consultas tenga la información más fresca sobre la distribución de los datos.
* **Limpieza**: Mantiene el almacenamiento bajo control aplicando una política de retención clara.



Script:

**Scripts de Backup con Compresión**

La compresión reduce drásticamente el tamaño del archivo (a menudo entre un 50% y 80%), lo que hace que los backups sean más rápidos y ocupen menos disco. Solo necesitas agregar la opción WITH COMPRESSION.

USE master;

GO

PRINT '--- 1. Generando Backups Comprimidos ---';

-- A. Backup FULL (Retención deseada: 4 semanas)

-- Nota: 'COMPRESSION' es la clave aquí.

BACKUP DATABASE [QhatuPERU]

TO DISK = 'C:\practicas\bds\full\QhatuPERU\_FULL\_Comprimido.bak'

WITH FORMAT, COMPRESSION, NAME = 'Full Backup Comprimido', STATS = 20;

-- B. Backup DIFERENCIAL (Retención deseada: 7 días)

BACKUP DATABASE [QhatuPERU]

TO DISK = 'C:\practicas\bds\diff\QhatuPERU\_DIFF\_Comprimido.dif'

WITH DIFFERENTIAL, COMPRESSION, NAME = 'Diff Backup Comprimido', STATS = 20;

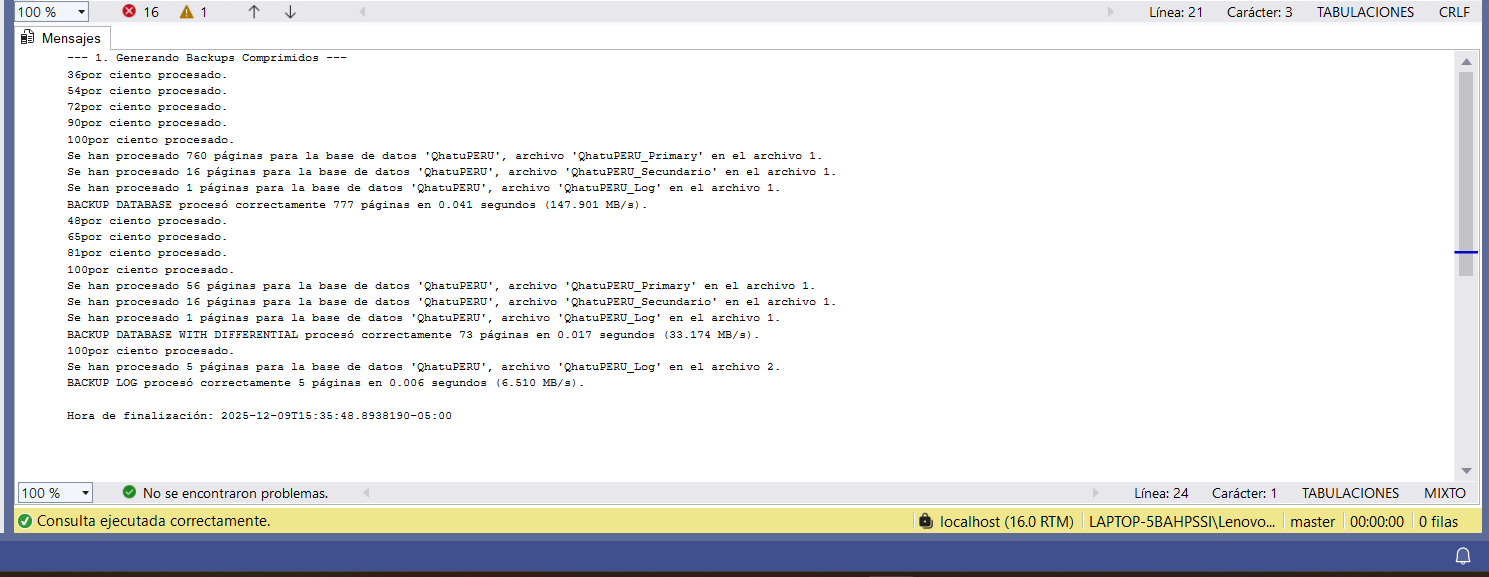
-- C. Backup de LOG (Retención deseada: 48 horas)

BACKUP LOG [QhatuPERU]

TO DISK = 'C:\practicas\bds\log\QhatuPERU\_LOG\_Comprimido.trn'

WITH COMPRESSION, NAME = 'Log Backup Comprimido', STATS = 20;

GO



**Script de Limpieza (Políticas de Retención Inteligentes)**

Este es el "cerebro" del ejercicio. En lugar de borrar todo lo que tenga X días, este script discrimina por **tipo de backup** (type en msdb) para aplicar la regla correcta a cada uno.

USE master;

GO

PRINT '--- 2. Ejecutando Limpieza según Políticas de Retención ---';

DECLARE @RutaArchivo NVARCHAR(512);

DECLARE @ComandoBorrar NVARCHAR(1000);

-- Cursor Inteligente: Selecciona archivos que CUMPLEN su criterio de eliminación

-- type 'D' = Full, 'I' = Diferencial, 'L' = Log

DECLARE CursorLimpieza CURSOR LOCAL FOR

SELECT DISTINCT bmf.physical\_device\_name

FROM msdb.dbo.backupset bs

INNER JOIN msdb.dbo.backupmediafamily bmf ON bs.media\_set\_id = bmf.media\_set\_id

WHERE bs.database\_name = 'QhatuPERU'

AND (

-- Regla 1: Borrar FULL ('D') mas antiguos de 4 semanas (28 días)

(bs.type = 'D' AND bs.backup\_finish\_date < DATEADD(week, -4, GETDATE()))

OR

-- Regla 2: Borrar DIFF ('I') mas antiguos de 7 días

(bs.type = 'I' AND bs.backup\_finish\_date < DATEADD(day, -7, GETDATE()))

OR

-- Regla 3: Borrar LOG ('L') mas antiguos de 48 horas

(bs.type = 'L' AND bs.backup\_finish\_date < DATEADD(hour, -48, GETDATE()))

)

AND bmf.physical\_device\_name NOT LIKE 'NULL';

OPEN CursorLimpieza;

FETCH NEXT FROM CursorLimpieza INTO @RutaArchivo;

WHILE @@FETCH\_STATUS = 0

BEGIN

-- Construir comando de borrado

SET @ComandoBorrar = 'DEL /F /Q "' + @RutaArchivo + '"';

PRINT 'Eliminando archivo caducado: ' + @RutaArchivo;

-- Intentar borrar archivo físico

BEGIN TRY

EXEC master..xp\_cmdshell @ComandoBorrar, NO\_OUTPUT;

END TRY

BEGIN CATCH

PRINT ' - Archivo no encontrado en disco (ya borrado o movido).';

END CATCH

FETCH NEXT FROM CursorLimpieza INTO @RutaArchivo;

END

CLOSE CursorLimpieza;

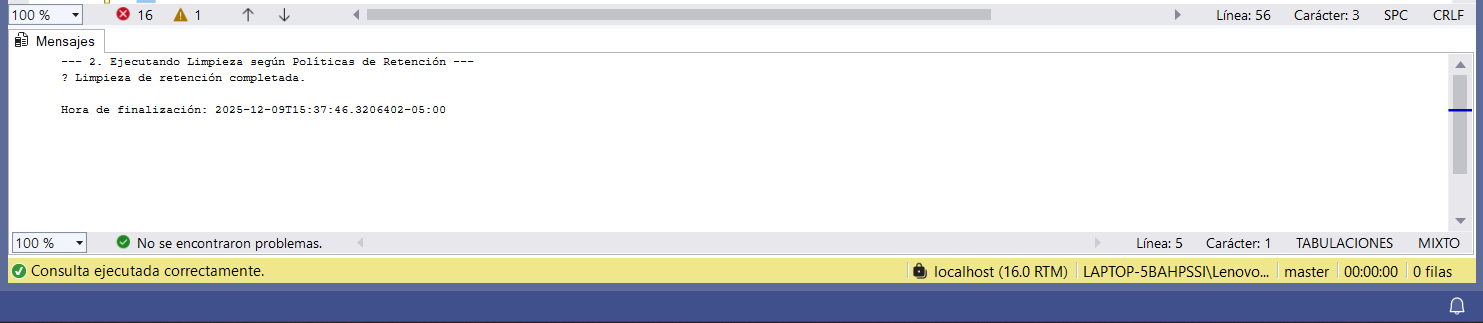
DEALLOCATE CursorLimpieza;

-- Limpieza final de metadatos en MSDB (borramos historial viejo general, ej. > 4 semanas)

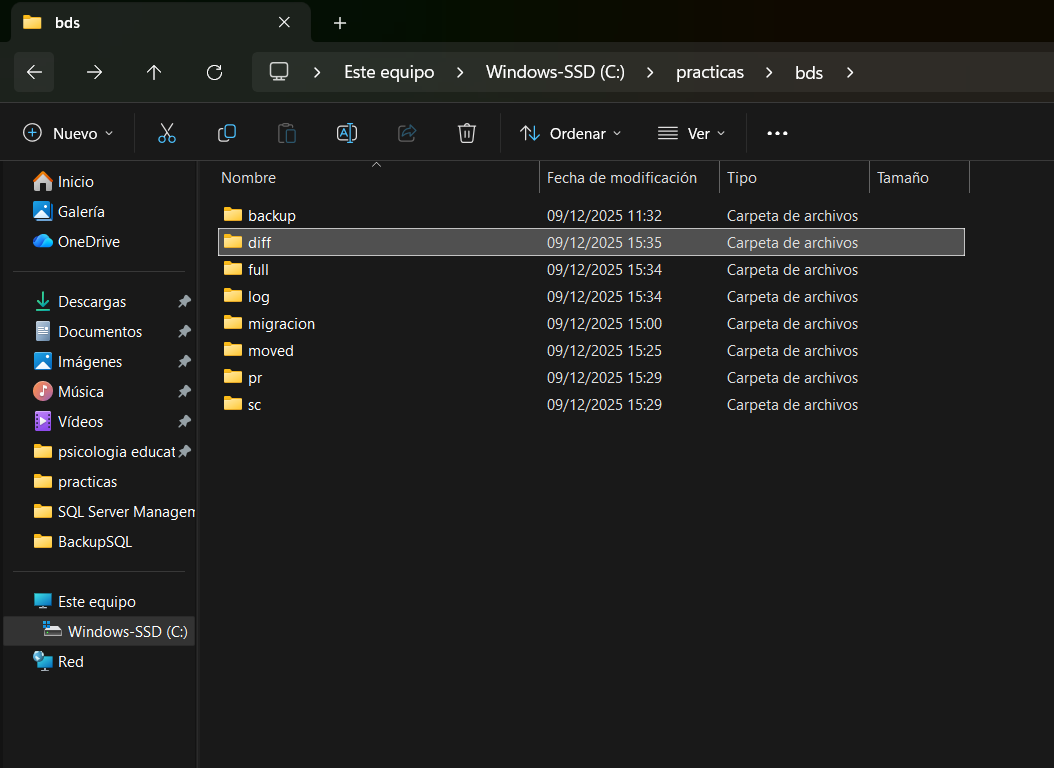
EXEC msdb.dbo.sp\_delete\_backuphistory @oldest\_date = '2025-11-01'; -- Ajusta esta fecha dinámicamente en producción (ej. DATEADD(week, -4, GETDATE()))

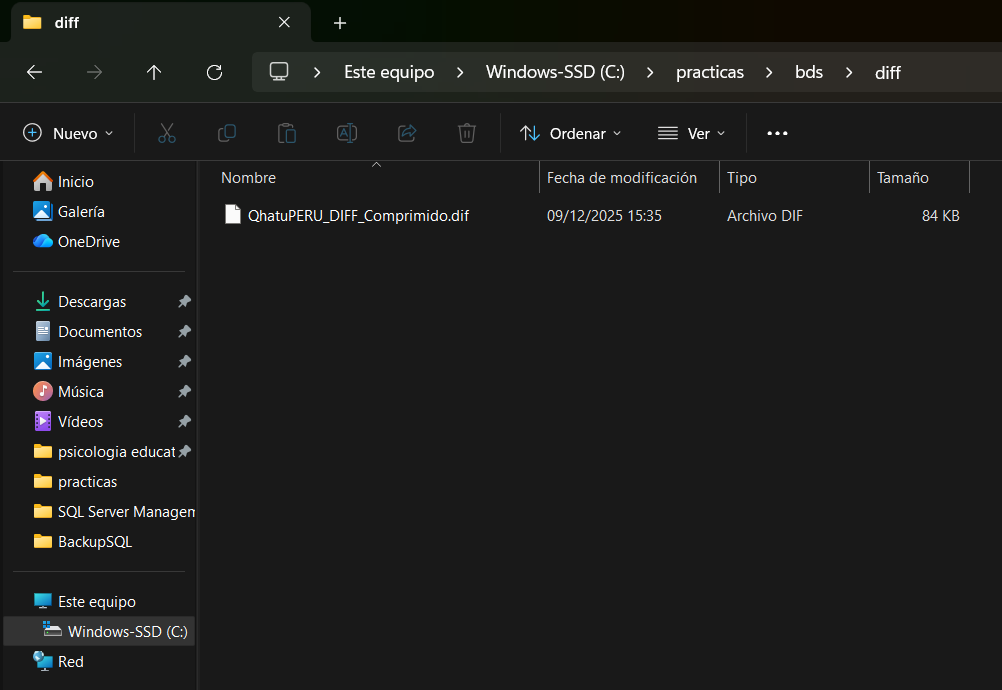
PRINT '✅ Limpieza de retención completada.';

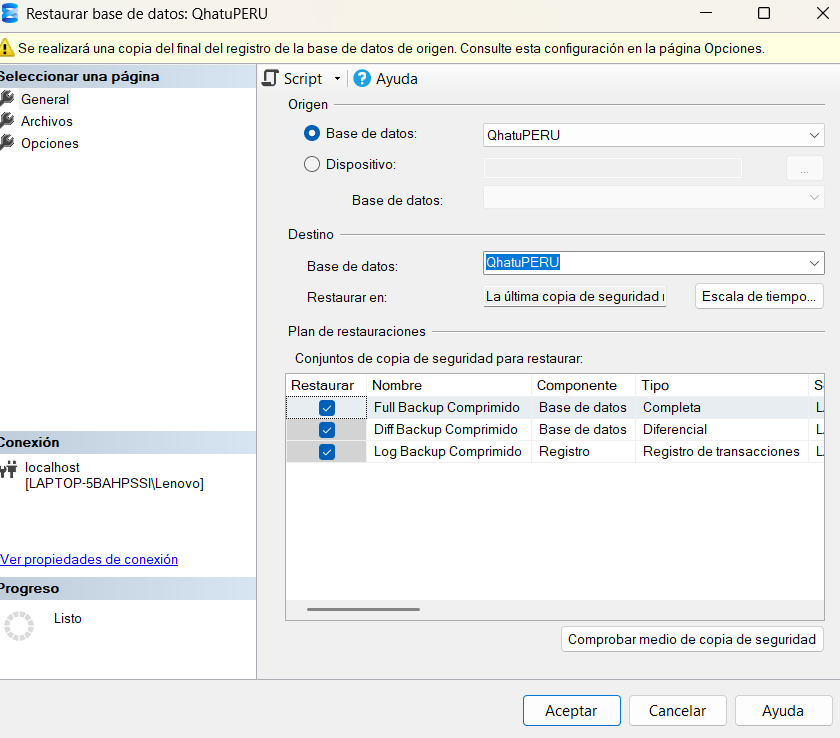
GO



Justificación:

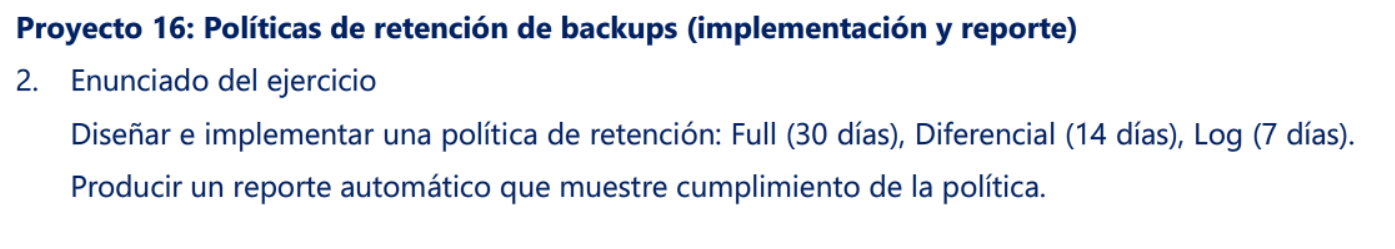






Buenas Prácticas:

* **Compresión de Backups:** Es una práctica estándar moderna. Ahorra costos de almacenamiento y reduce el tiempo de transferencia por red (menos datos que mover), aunque utiliza un poco más de CPU durante la creación.
* **Retención Granular (Tiered Retention):** No todos los backups valen lo mismo. Los Logs pierden utilidad muy rápido (en 48h suelen ser irrelevantes si tienes un FULL nuevo), mientras que los FULL deben guardarse más tiempo (semanas o meses) por cumplimiento legal o auditoría. Definir reglas diferentes optimiza el uso del disco.



Script:

**Implementación de la Política (Script de Limpieza)**

Este script es el "recolector de basura". Busca archivos que hayan excedido su tiempo de vida específico según su tipo y los elimina físicamente.

USE master;

GO

PRINT '--- INICIANDO APLICACIÓN DE POLÍTICAS DE RETENCIÓN ---';

DECLARE @RutaArchivo NVARCHAR(512);

DECLARE @CmdBorrar NVARCHAR(1000);

-- Definimos las fechas de corte según el requerimiento

DECLARE @LimiteFull DATETIME = DATEADD(DAY, -30, GETDATE()); -- 30 Días

DECLARE @LimiteDiff DATETIME = DATEADD(DAY, -14, GETDATE()); -- 14 Días

DECLARE @LimiteLog DATETIME = DATEADD(DAY, -7, GETDATE()); -- 7 Días

-- Cursor Inteligente: Selecciona archivos CADUCADOS según su tipo

DECLARE CursorRetencion CURSOR LOCAL FOR

SELECT DISTINCT bmf.physical\_device\_name

FROM msdb.dbo.backupset bs

INNER JOIN msdb.dbo.backupmediafamily bmf ON bs.media\_set\_id = bmf.media\_set\_id

WHERE bs.database\_name = 'QhatuPERU'

AND (

(bs.type = 'D' AND bs.backup\_finish\_date < @LimiteFull) -- Full vencidos

OR

(bs.type = 'I' AND bs.backup\_finish\_date < @LimiteDiff) -- Diff vencidos

OR

(bs.type = 'L' AND bs.backup\_finish\_date < @LimiteLog) -- Log vencidos

)

AND bmf.physical\_device\_name NOT LIKE 'NULL';

OPEN CursorRetencion;

FETCH NEXT FROM CursorRetencion INTO @RutaArchivo;

WHILE @@FETCH\_STATUS = 0

BEGIN

SET @CmdBorrar = 'DEL /F /Q "' + @RutaArchivo + '"';

BEGIN TRY

-- Eliminación Física

EXEC master..xp\_cmdshell @CmdBorrar, NO\_OUTPUT;

PRINT '🗑️ Archivo eliminado por política: ' + @RutaArchivo;

END TRY

BEGIN CATCH

PRINT '⚠️ No se encontró el archivo (posiblemente ya borrado): ' + @RutaArchivo;

END CATCH

FETCH NEXT FROM CursorRetencion INTO @RutaArchivo;

END

CLOSE CursorRetencion;

DEALLOCATE CursorRetencion;

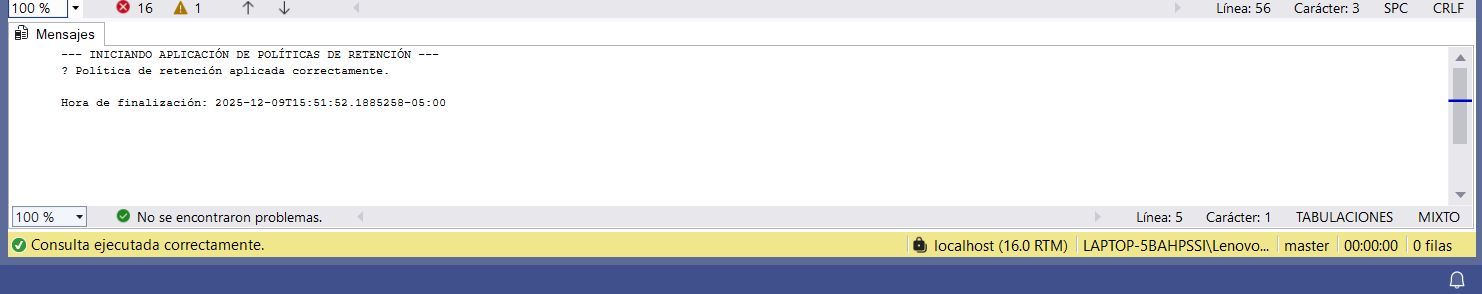
-- Limpieza de Metadatos (Historial en MSDB)

-- Borramos el historial más antiguo que nuestra política más larga (30 días)

EXEC msdb.dbo.sp\_delete\_backuphistory @oldest\_date = @LimiteFull;

PRINT '✅ Política de retención aplicada correctamente.';

GO



**Reporte Automático de Cumplimiento**

Este script es para el auditor o el jefe. Genera una tabla resumen que muestra qué backups tienes y si estás cumpliendo la promesa de retención.

USE master;

GO

PRINT '--- REPORTE DE CUMPLIMIENTO DE RETENCIÓN ---';

SELECT

CASE bs.type

WHEN 'D' THEN 'FULL (Completo)'

WHEN 'I' THEN 'DIFF (Diferencial)'

WHEN 'L' THEN 'LOG (Transacciones)'

END AS [Tipo de Backup],

COUNT(\*) AS [Cantidad Archivos],

MIN(bs.backup\_finish\_date) AS [Archivo Más Antiguo],

MAX(bs.backup\_finish\_date) AS [Archivo Más Reciente],

-- Columna de Validación Automática

CASE

WHEN bs.type = 'D' AND MIN(bs.backup\_finish\_date) < DATEADD(DAY, -30, GETDATE()) THEN '❌ INCUMPLE (>30 días)'

WHEN bs.type = 'I' AND MIN(bs.backup\_finish\_date) < DATEADD(DAY, -14, GETDATE()) THEN '❌ INCUMPLE (>14 días)'

WHEN bs.type = 'L' AND MIN(bs.backup\_finish\_date) < DATEADD(DAY, -7, GETDATE()) THEN '❌ INCUMPLE (>7 días)'

ELSE '✅ CUMPLE (Dentro de Política)'

END AS [Estado de Cumplimiento]

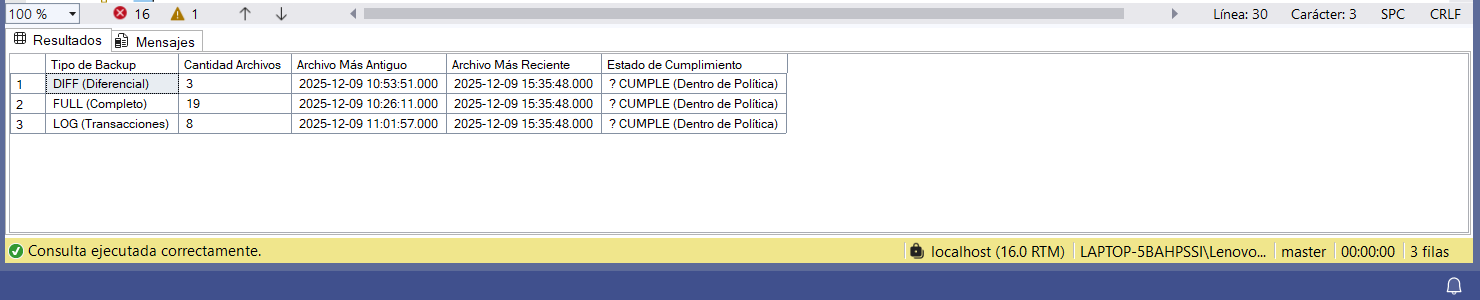
FROM msdb.dbo.backupset bs

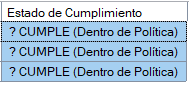
WHERE bs.database\_name = 'QhatuPERU'

GROUP BY bs.type

ORDER BY [Tipo de Backup];

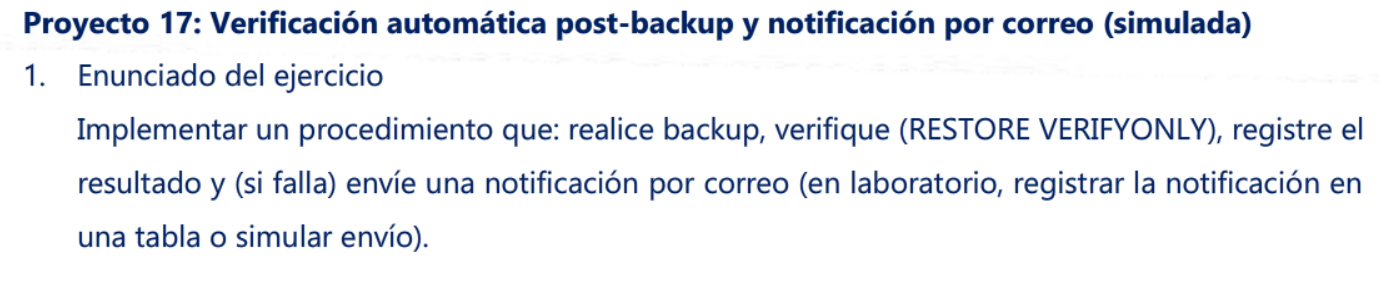
GO





Buenas Prácticas:

Este ejercicio implementa la **Gobernanza del Ciclo de Vida de Datos (DLM)**. No basta con acumular copias de seguridad; es necesario gestionar su eliminación para controlar costos de almacenamiento y cumplir con normativas legales (como GDPR o normas bancarias) que a menudo exigen borrar datos después de cierto tiempo. El **Reporte de Cumplimiento** automatizado es la herramienta clave para auditorías, transformando registros técnicos oscuros (msdb) en información de negocio clara ("Cumple/No Cumple").



Script:

**Configuración: Tabla de Correos Simulados**

Primero, creamos el "Buzón de Salida" donde se guardarán las alertas que el sistema intentaría enviar.

USE [QhatuPERU];

GO

IF OBJECT\_ID('dbo.Alertas\_Correo\_Simulado') IS NULL

BEGIN

CREATE TABLE dbo.Alertas\_Correo\_Simulado (

ID INT IDENTITY(1,1) PRIMARY KEY,

FechaAlerta DATETIME DEFAULT GETDATE(),

Destinatario VARCHAR(100) DEFAULT 'admin@qhatuperu.com',

Asunto VARCHAR(200),

Mensaje VARCHAR(MAX),

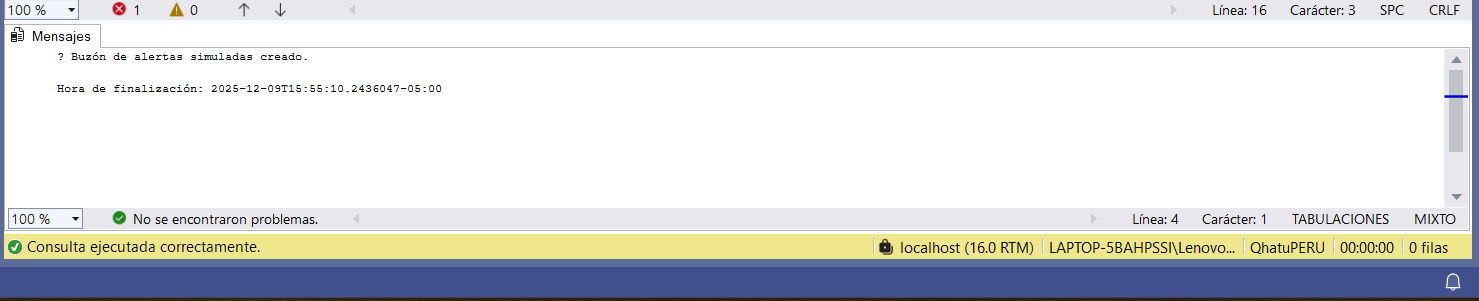
Prioridad VARCHAR(20)

);

PRINT '✅ Buzón de alertas simuladas creado.';

END

GO



**Backup con Verificación y Alerta Automática**

Este script intenta hacer el backup y verificarlo. Si la verificación falla (simulado en el bloque CATCH), dispara la alerta a la tabla.

DECLARE @RutaBackup NVARCHAR(500);

DECLARE @FechaTexto NVARCHAR(20);

DECLARE @AsuntoMail NVARCHAR(200);

DECLARE @CuerpoMail NVARCHAR(MAX);

-- Generar ruta única

SELECT @FechaTexto = REPLACE(REPLACE(CONVERT(VARCHAR(19), GETDATE(), 120), ':', ''), '-', '');

SET @RutaBackup = 'C:\practicas\bds\full\QhatuPERU\_AutoVerify\_' + @FechaTexto + '.bak';

PRINT '--- Iniciando Proceso de Respaldo y Verificación ---';

BEGIN TRY

-- 1. REALIZAR BACKUP

BACKUP DATABASE [QhatuPERU]

TO DISK = @RutaBackup

WITH FORMAT, NAME = 'Backup Auto-Verificado';

PRINT '✅ Backup generado exitosamente.';

-- 2. VERIFICAR INTEGRIDAD (El momento de la verdad)

RESTORE VERIFYONLY FROM DISK = @RutaBackup;

PRINT '✅ Verificación (VerifyOnly) completada: El archivo está sano.';

-- (Opcional) Registrar el éxito en un log normal, pero no enviamos alerta porque todo está bien.

END TRY

BEGIN CATCH

-- 3. GESTIÓN DEL FALLO (Disparador de Alerta)

PRINT '❌ ¡ERROR DETECTADO! Iniciando protocolo de notificación...';

SET @AsuntoMail = 'URGENTE: Fallo de Backup en QhatuPERU';

SET @CuerpoMail = 'El sistema detectó un error al verificar el backup: ' + @RutaBackup +

'. Detalles del error SQL: ' + ERROR\_MESSAGE();

-- Simulamos el envío del correo (sp\_send\_dbmail) insertando en la tabla

INSERT INTO dbo.Alertas\_Correo\_Simulado (Asunto, Mensaje, Prioridad)

VALUES (@AsuntoMail, @CuerpoMail, 'ALTA');

PRINT '📧 Notificación de alerta enviada al administrador.';

END CATCH

GO



Fallo provocado:  
-- SCRIPT DE PRUEBA DE FALLO (Ejecútalo para ver la alerta)

BEGIN TRY

-- Intentamos verificar un archivo fantasma

RESTORE VERIFYONLY FROM DISK = 'C:\Ruta\Falsa\NoExiste.bak';

END TRY

BEGIN CATCH

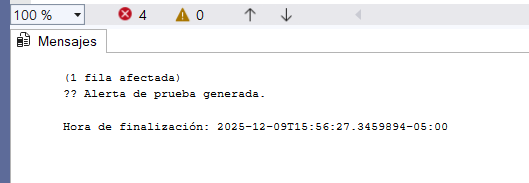
INSERT INTO dbo.Alertas\_Correo\_Simulado (Asunto, Mensaje, Prioridad)

VALUES ('PRUEBA DE FALLO', 'Error simulado: ' + ERROR\_MESSAGE(), 'MEDIA');

PRINT '📧 Alerta de prueba generada.';

END CATCH

GO



Justificación:

SELECT \* FROM [QhatuPERU].dbo.Alertas\_Correo\_Simulado

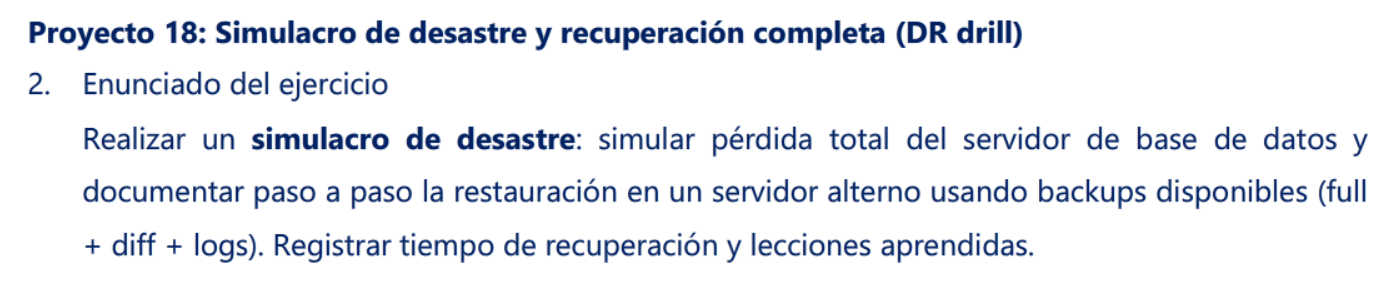
ORDER BY FechaAlerta DESC;



Esto confirma que, si ocurre un desastre real a las 3:00 AM, el sistema dejará el registro listo para ser enviado.

Buenas Prácticas:

Este ejercicio implementa la **Monitorización Proactiva y Alertado**. En administración de bases de datos, un "backup fallido silencioso" es el peor escenario posible. La buena práctica dicta que **todo fallo crítico debe generar una notificación push** (correo, SMS, Slack) hacia el DBA, en lugar de esperar a que alguien revise los logs manualmente. Simular esto con una tabla es el paso previo a configurar **Database Mail** en SQL Server para envíos reales.



Script:

**Generación de los Backups (Evidencia)**

Primero, generamos los archivos (Full, Diff y Log) que "salvaremos" del servidor quemado

USE master;

GO

ALTER DATABASE [QhatuPERU] SET RECOVERY FULL;

GO

PRINT '--- 1. GENERANDO ARTEFACTOS DE RESPALDO (Los archivos que salvarás) ---';

-- A. Backup FULL (Base)

BACKUP DATABASE [QhatuPERU]

TO DISK = 'C:\practicas\bds\full\QhatuPERU\_DR\_Full.bak'

WITH FORMAT, NAME = 'Full DR Base';

-- Simular cambios...

WAITFOR DELAY '00:00:02';

-- B. Backup DIFF (Cambios intermedios)

BACKUP DATABASE [QhatuPERU]

TO DISK = 'C:\practicas\bds\diff\QhatuPERU\_DR\_Diff.dif'

WITH FORMAT, NAME = 'Diff DR Delta';

-- Simular más cambios...

WAITFOR DELAY '00:00:02';

-- C. Backup LOG (Últimas transacciones)

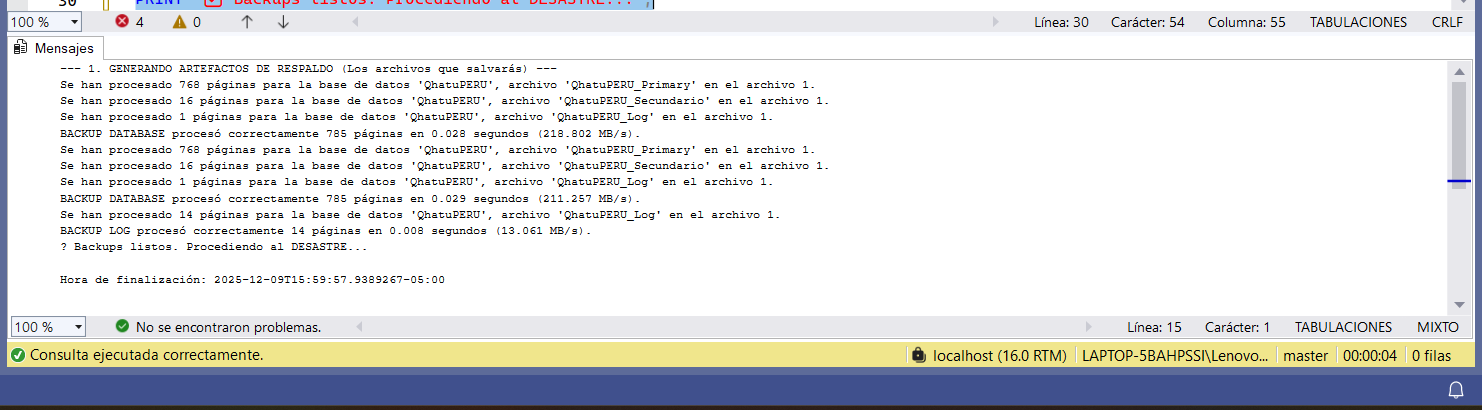
BACKUP LOG [QhatuPERU]

TO DISK = 'C:\practicas\bds\log\QhatuPERU\_DR\_Log.trn'

WITH FORMAT, NAME = 'Log DR Final';

GO

PRINT '✅ Backups listos. Procediendo al DESASTRE...';



**El Desastre (Simulación de Pérdida Total)**

Aquí simulamos que el servidor original explotó. Borraremos la base de datos QhatuPERU por completo.

USE master;

GO

PRINT '--- 2. SIMULANDO PÉRDIDA TOTAL DEL SERVIDOR ---';

IF EXISTS (SELECT name FROM sys.databases WHERE name = N'QhatuPERU')

BEGIN

ALTER DATABASE [QhatuPERU] SET SINGLE\_USER WITH ROLLBACK IMMEDIATE;

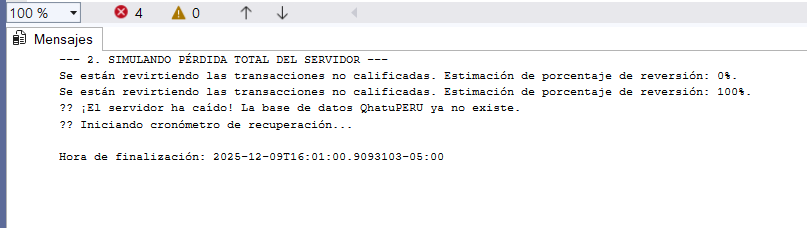
DROP DATABASE [QhatuPERU];

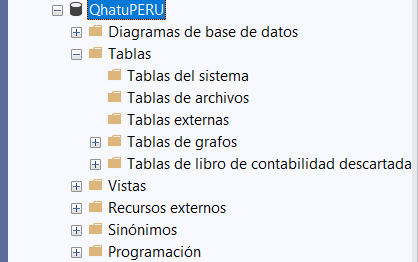
END

PRINT '🔥 ¡El servidor ha caído! La base de datos QhatuPERU ya no existe.';

PRINT '🕒 Iniciando cronómetro de recuperación...';

GO





**Recuperación en "Servidor Alterno" (El Rescate)**

Ahora actuamos como el DBA en el nuevo servidor. Restauraremos la base de datos con el nombre QhatuPERU\_Recovered (simulando el servidor alterno) y mediremos cuánto tiempo nos tomó.

USE master;

GO

-- Variables para medir el RTO (Recovery Time Objective)

DECLARE @TiempoInicio DATETIME = GETDATE();

DECLARE @TiempoFin DATETIME;

PRINT '--- 3. INICIANDO PROTOCOLO DE RECUPERACIÓN (DR) ---';

-- A. Restaurar FULL con NORECOVERY (Cimientos)

RESTORE DATABASE [QhatuPERU\_Recovered] -- Nombre en el "Nuevo Servidor"

FROM DISK = 'C:\practicas\bds\full\QhatuPERU\_DR\_Full.bak'

WITH

NORECOVERY,

-- Movemos los archivos para simular la estructura del nuevo disco

MOVE 'QhatuPERU\_Primary' TO 'C:\practicas\bds\moved\QhatuPERU\_DR.mdf',

MOVE 'QhatuPERU\_Secundario' TO 'C:\practicas\bds\moved\QhatuPERU\_DR\_Sec.ndf',

MOVE 'QhatuPERU\_Log' TO 'C:\practicas\bds\moved\QhatuPERU\_DR\_Log.ldf';

-- B. Restaurar DIFF con NORECOVERY (Paredes)

RESTORE DATABASE [QhatuPERU\_Recovered]

FROM DISK = 'C:\practicas\bds\diff\QhatuPERU\_DR\_Diff.dif'

WITH NORECOVERY;

-- C. Restaurar LOG con RECOVERY (Techo y Acabados -> ¡Online!)

RESTORE LOG [QhatuPERU\_Recovered]

FROM DISK = 'C:\practicas\bds\log\QhatuPERU\_DR\_Log.trn'

WITH RECOVERY;

-- Medir tiempo final

SET @TiempoFin = GETDATE();

PRINT '✅ ¡SERVICIO RESTAURADO EN SERVIDOR ALTERNO!';

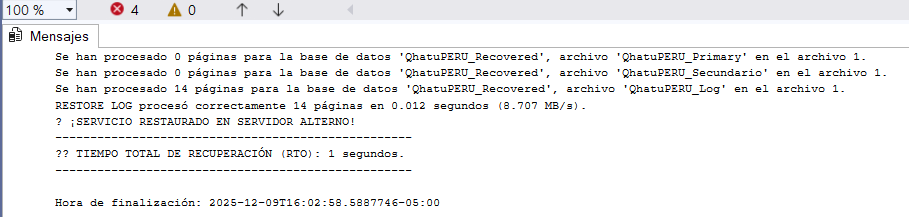
PRINT '---------------------------------------------------';

PRINT '⏱️ TIEMPO TOTAL DE RECUPERACIÓN (RTO): ' +

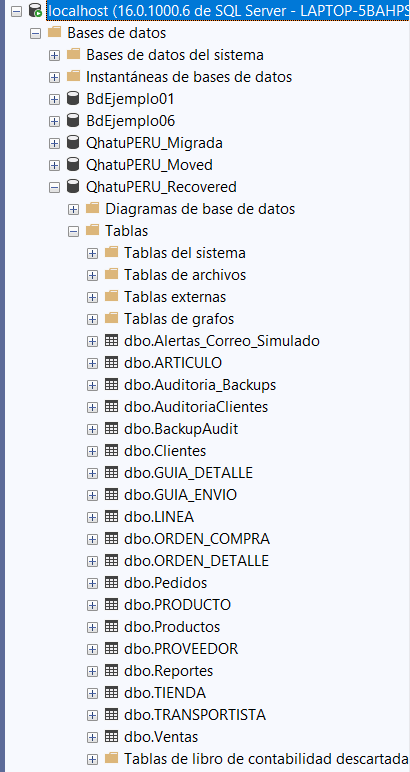
CAST(DATEDIFF(SECOND, @TiempoInicio, @TiempoFin) AS VARCHAR) + ' segundos.';

PRINT '---------------------------------------------------';

GO

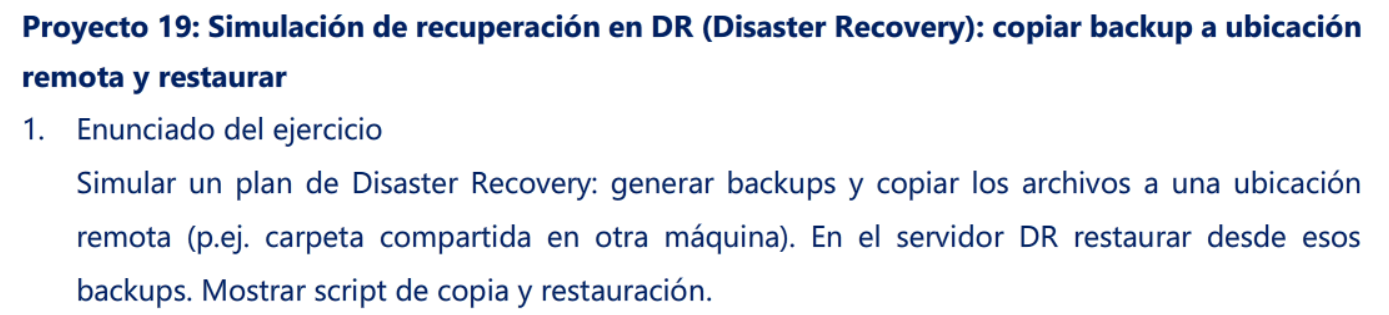


Justificación:



Buenas Prácticas:

* **Simulacros Reales:** No basta con tener backups; hay que "romper" cosas controladamente para entrenar la memoria muscular y reducir el pánico cuando ocurra un desastre real.
* **Medición del RTO:** Registrar el tiempo que toma restaurar (DATEDIFF) es vital para saber si cumples con los acuerdos de nivel de servicio (SLA) del negocio.
* **Uso de Cadena Completa:** Demostramos que la combinación FULL + DIFF + LOG es la estrategia más robusta para reconstruir la historia completa de la base de datos hasta el último segundo.



Script:

En el ejercicio anterior se elimina la base de datos y pasa a usarse la restauración Recovered, para no compliar el procedimiento de los demás proyectos se regresa a la “normalidad” la base de datos QhatuPERU:

USE master;

GO

-- Verificamos si existe la recuperada y NO existe la original

IF EXISTS (SELECT name FROM sys.databases WHERE name = N'QhatuPERU\_Recovered')

AND NOT EXISTS (SELECT name FROM sys.databases WHERE name = N'QhatuPERU')

BEGIN

-- Poner en SINGLE\_USER para evitar bloqueos durante el cambio de nombre

ALTER DATABASE [QhatuPERU\_Recovered] SET SINGLE\_USER WITH ROLLBACK IMMEDIATE;

-- El comando mágico para renombrar

ALTER DATABASE [QhatuPERU\_Recovered] MODIFY NAME = [QhatuPERU];

-- Volver a MULTI\_USER

ALTER DATABASE [QhatuPERU] SET MULTI\_USER;

PRINT '✅ Base de datos renombrada exitosamente. Todo ha vuelto a la normalidad.';

END

ELSE

BEGIN

PRINT '⚠️ No se requiere cambios (o la base QhatuPERU ya existe).';

END

GO

**Preparación (Crear carpeta remota simulada)**

Antes de ejecutar el código, crea manualmente esta carpeta en tu Windows:

* C:\practicas\bds\remote\_share\

USE master;

GO

PRINT '--- INICIO PROYECTO 19: REPLICACIÓN A SITIO REMOTO (DR) ---';

-- Declaramos variables

DECLARE @RutaLocal NVARCHAR(255) = 'C:\practicas\bds\full\QhatuPERU\_Para\_Remote.bak';

DECLARE @RutaRemota NVARCHAR(255) = 'C:\practicas\bds\remote\_share\QhatuPERU\_Para\_Remote.bak';

DECLARE @ComandoCopia NVARCHAR(500);

-- 1. GENERAR BACKUP EN SITIO LOCAL

PRINT '>> 1. Generando backup local...';

BACKUP DATABASE [QhatuPERU]

TO DISK = @RutaLocal

WITH FORMAT, NAME = 'Backup para Transferencia DR';

-- 2. SIMULAR COPIA A UBICACIÓN REMOTA (Copy over Network)

-- En la vida real, esto sería una ruta de red como \\ServidorDR\Backups\

PRINT '>> 2. Copiando archivo a ubicación remota (Simulada)...';

SET @ComandoCopia = 'COPY /Y "' + @RutaLocal + '" "' + @RutaRemota + '"';

BEGIN TRY

-- Usamos xp\_cmdshell para ejecutar el comando de copia de Windows

EXEC master..xp\_cmdshell @ComandoCopia, NO\_OUTPUT;

PRINT '✅ Archivo copiado exitosamente a: ' + @RutaRemota;

END TRY

BEGIN CATCH

PRINT '❌ Error al copiar el archivo. Verifica que la carpeta exista.';

END CATCH

-- 3. RESTAURAR EN EL "SITIO DR" (Simulado en el mismo server con otro nombre)

PRINT '>> 3. Simulando restauración en Servidor de Contingencia...';

-- Restauramos como 'QhatuPERU\_DR\_Site'

RESTORE DATABASE [QhatuPERU\_DR\_Site]

FROM DISK = @RutaRemota -- ¡OJO! Restauramos desde la copia REMOTA, no la local

WITH

RECOVERY,

-- Movemos los archivos para no chocar con la original

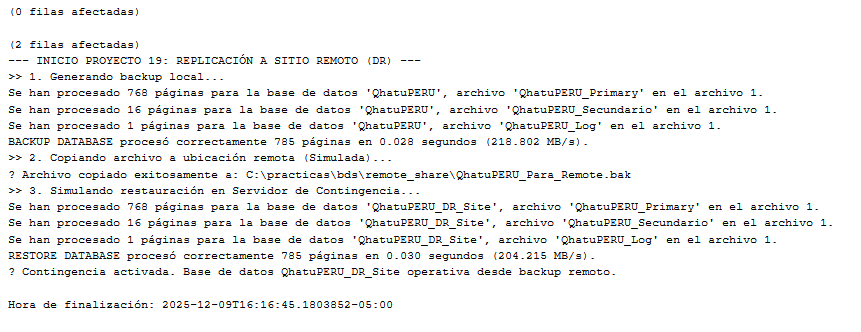
MOVE 'QhatuPERU\_Primary' TO 'C:\practicas\bds\remote\_share\Qhatu\_DR.mdf',

MOVE 'QhatuPERU\_Secundario' TO 'C:\practicas\bds\remote\_share\Qhatu\_DR\_Sec.ndf',

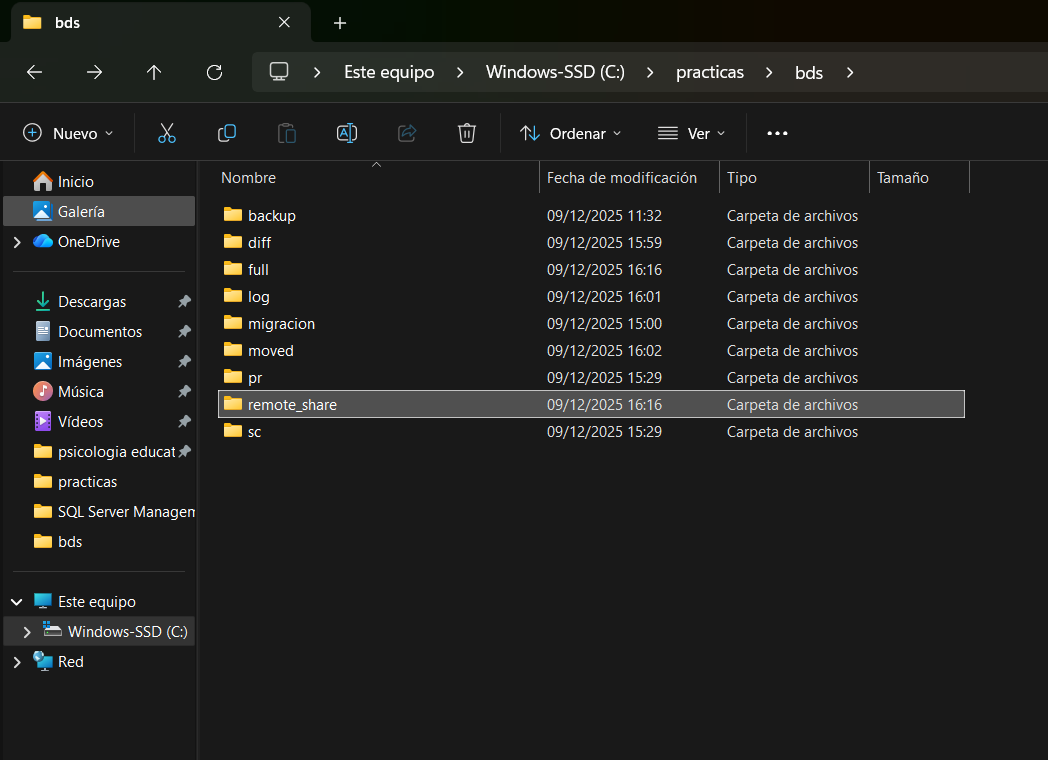
MOVE 'QhatuPERU\_Log' TO 'C:\practicas\bds\remote\_share\Qhatu\_DR\_Log.ldf';

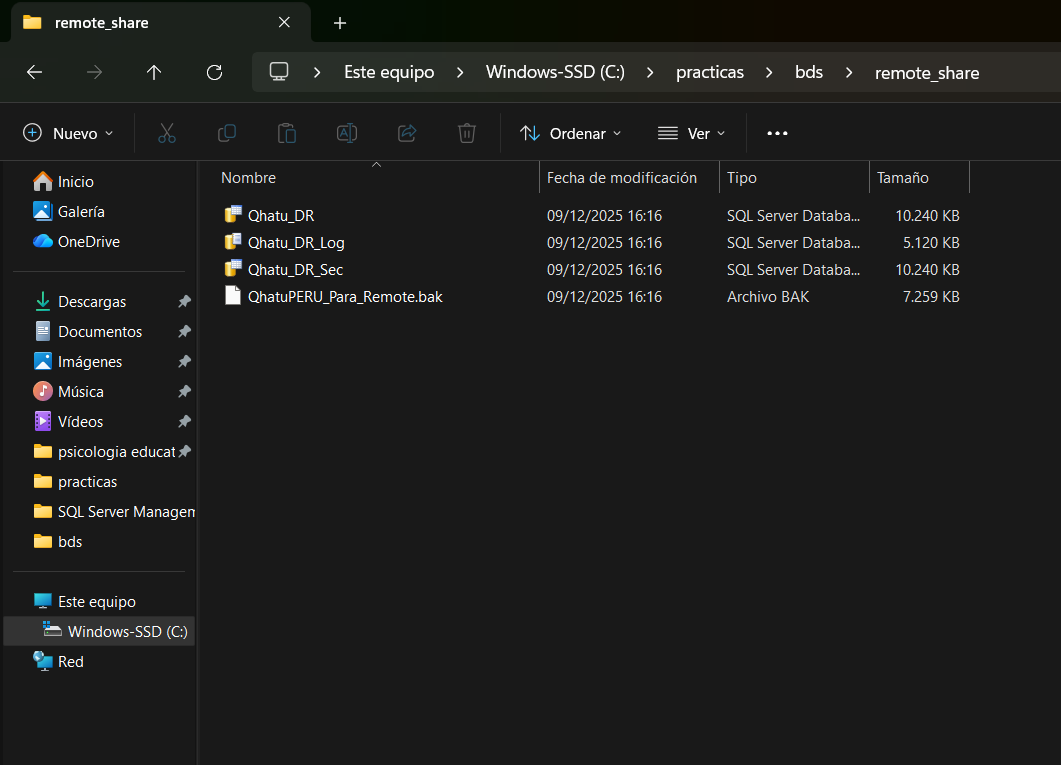
PRINT '✅ Contingencia activada. Base de datos QhatuPERU\_DR\_Site operativa desde backup remoto.';

GO



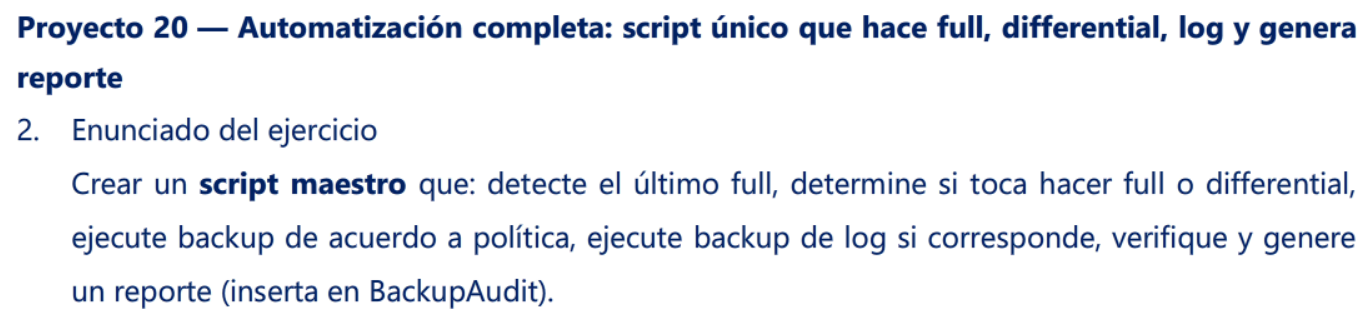
Justificación:





Buenas Prácticas:

* **Regla 3-2-1 de Backups:** Este ejercicio aplica la regla fundamental: tener al menos **3** copias de los datos, en **2** medios diferentes, y **1** de ellas fuera del sitio (Off-site). La copia a remote\_share simula esa copia "fuera del sitio".
* **Validación de Transporte:** Al restaurar desde la copia remota (y no desde la local), verificamos no solo que el backup sirve, sino que el archivo **no se corrompió durante la transferencia** por la red.
* **Normalización Post-Desastre:** El paso previo de renombrar la base de datos es una práctica estándar de DBA; después de un incidente, el objetivo es devolver el sistema a su estado original (nombres estándar) para que las aplicaciones y usuarios no noten la diferencia.

  
Script:

Este script consulta la "memoria" de SQL Server (msdb) para ver cuándo fue el último backup FULL.

* **Regla:** Si el último FULL tiene más de 7 días (o no existe), hace un **FULL**.
* **Regla:** Si el último FULL es reciente (menos de 7 días), hace un **DIFERENCIAL**.
* **Siempre:** Hace un backup de **LOG** y registra todo en la tabla de auditoría.

USE [QhatuPERU];

GO

-- Aseguramos que la tabla de auditoría exista (del Proyecto 2)

IF OBJECT\_ID('dbo.BackupAudit') IS NULL

BEGIN

CREATE TABLE dbo.BackupAudit (

ID INT IDENTITY(1,1) PRIMARY KEY,

FechaVerificacion DATETIME DEFAULT GETDATE(),

NombreArchivo NVARCHAR(500),

Estado VARCHAR(20),

Mensaje NVARCHAR(MAX)

);

END

GO

PRINT '--- INICIANDO SCRIPT MAESTRO DE BACKUP INTELIGENTE ---';

DECLARE @UltimoFull DATETIME;

DECLARE @TipoBackupData VARCHAR(10);

DECLARE @RutaArchivoData NVARCHAR(500);

DECLARE @RutaArchivoLog NVARCHAR(500);

DECLARE @FechaTexto NVARCHAR(20);

DECLARE @MensajeError NVARCHAR(MAX) = 'OK';

DECLARE @EstadoFinal VARCHAR(20) = 'EXITOSO';

-- Generar marca de tiempo para archivos

SELECT @FechaTexto = REPLACE(REPLACE(CONVERT(VARCHAR(19), GETDATE(), 120), ':', ''), '-', '');

-- 1. CEREBRO: ¿Cuándo fue el último Backup FULL?

SELECT @UltimoFull = MAX(backup\_finish\_date)

FROM msdb.dbo.backupset

WHERE database\_name = 'QhatuPERU' AND type = 'D'; -- 'D' = Database (Full)

-- 2. DECISIÓN: ¿Toca FULL o DIFF?

IF @UltimoFull IS NULL OR @UltimoFull < DATEADD(DAY, -7, GETDATE())

BEGIN

SET @TipoBackupData = 'FULL';

SET @RutaArchivoData = 'C:\practicas\bds\full\QhatuPERU\_Auto\_FULL\_' + @FechaTexto + '.bak';

END

ELSE

BEGIN

SET @TipoBackupData = 'DIFF';

SET @RutaArchivoData = 'C:\practicas\bds\diff\QhatuPERU\_Auto\_DIFF\_' + @FechaTexto + '.dif';

END

PRINT '>> Decisión del Sistema: Se realizará un backup ' + @TipoBackupData;

-- 3. EJECUCIÓN: Backup de DATOS (Full o Diff según decisión)

BEGIN TRY

IF @TipoBackupData = 'FULL'

BACKUP DATABASE [QhatuPERU] TO DISK = @RutaArchivoData WITH FORMAT, NAME = 'Auto Full';

ELSE

BACKUP DATABASE [QhatuPERU] TO DISK = @RutaArchivoData WITH DIFFERENTIAL, NAME = 'Auto Diff';

-- Verificación inmediata

RESTORE VERIFYONLY FROM DISK = @RutaArchivoData;

-- Reportar éxito en Auditoría

INSERT INTO dbo.BackupAudit (NombreArchivo, Estado, Mensaje)

VALUES (@RutaArchivoData, 'VALIDO', 'Backup ' + @TipoBackupData + ' generado y verificado por Script Maestro.');

PRINT '✅ Backup de Datos (' + @TipoBackupData + ') completado.';

END TRY

BEGIN CATCH

INSERT INTO dbo.BackupAudit (NombreArchivo, Estado, Mensaje)

VALUES (@RutaArchivoData, 'FALLO', ERROR\_MESSAGE());

PRINT '❌ Error en Backup de Datos.';

END CATCH

-- 4. EJECUCIÓN: Backup de LOG (Siempre, para mantener la cadena)

SET @RutaArchivoLog = 'C:\practicas\bds\log\QhatuPERU\_Auto\_LOG\_' + @FechaTexto + '.trn';

BEGIN TRY

BACKUP LOG [QhatuPERU] TO DISK = @RutaArchivoLog WITH NAME = 'Auto Log';

-- Verificación

RESTORE VERIFYONLY FROM DISK = @RutaArchivoLog;

INSERT INTO dbo.BackupAudit (NombreArchivo, Estado, Mensaje)

VALUES (@RutaArchivoLog, 'VALIDO', 'Backup LOG generado y verificado por Script Maestro.');

PRINT '✅ Backup de LOG completado.';

END TRY

BEGIN CATCH

INSERT INTO dbo.BackupAudit (NombreArchivo, Estado, Mensaje)

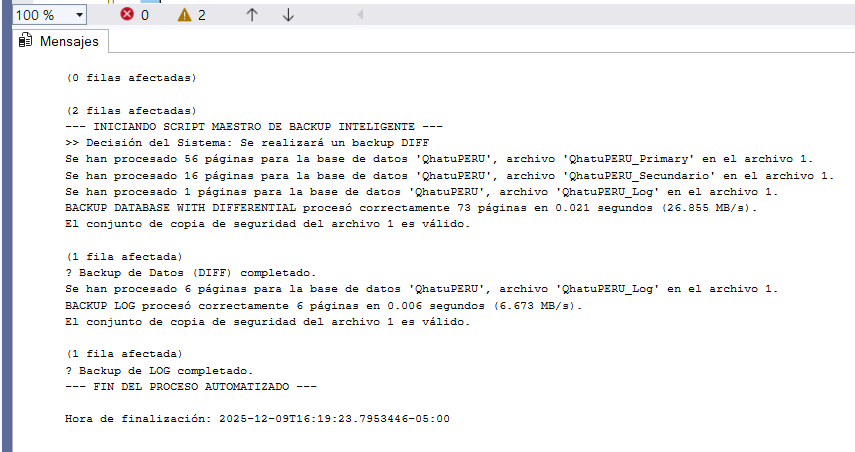
VALUES (@RutaArchivoLog, 'FALLO', ERROR\_MESSAGE());

PRINT '❌ Error en Backup de LOG.';

END CATCH

PRINT '--- FIN DEL PROCESO AUTOMATIZADO ---';

GO



Justificación:

SELECT \* FROM dbo.BackupAudit ORDER BY ID DESC;



Buenas Prácticas:

Este proyecto implementa la **Automatización Inteligente Basada en Políticas**. En lugar de tener múltiples Jobs rígidos (uno para Full el domingo, otro para Diff el lunes), utilizamos un solo script dinámico que consulta los metadatos del sistema (msdb) para determinar el estado actual de la protección de datos. Esto reduce el error humano, simplifica la administración (solo mantienes un script) y garantiza que, si un backup Full falla o se pierde, el sistema intentará generar uno nuevo automáticamente en la siguiente ejecución, manteniendo la cadena de recuperación siempre saludable (Self-Healing Backup Strategy).