UNIVERSIDAD PERUANA LOS ANDES FACULTAD DE INGENIERÍA

Asignatura: Base de Datos II

Docente: Dr. Raúl Enrique Fernández Bejarano

Alumno: SORIANO TIMOTEO Joel Kevin

Ciclo: V

huancayo-Perú-2025

Manual de practica Calificada Semana 12

Proyecto 1: Estrategia de Backup Completo Diario y Verificación Automática

1. Enunciado del Ejercicio

Implementar un script que haga un **backup FULL diario** de la base `QhatuPeru`, almacene el archivo con fecha, y **verifique la integridad del backup** (`RESTORE VERIFYONLY`).

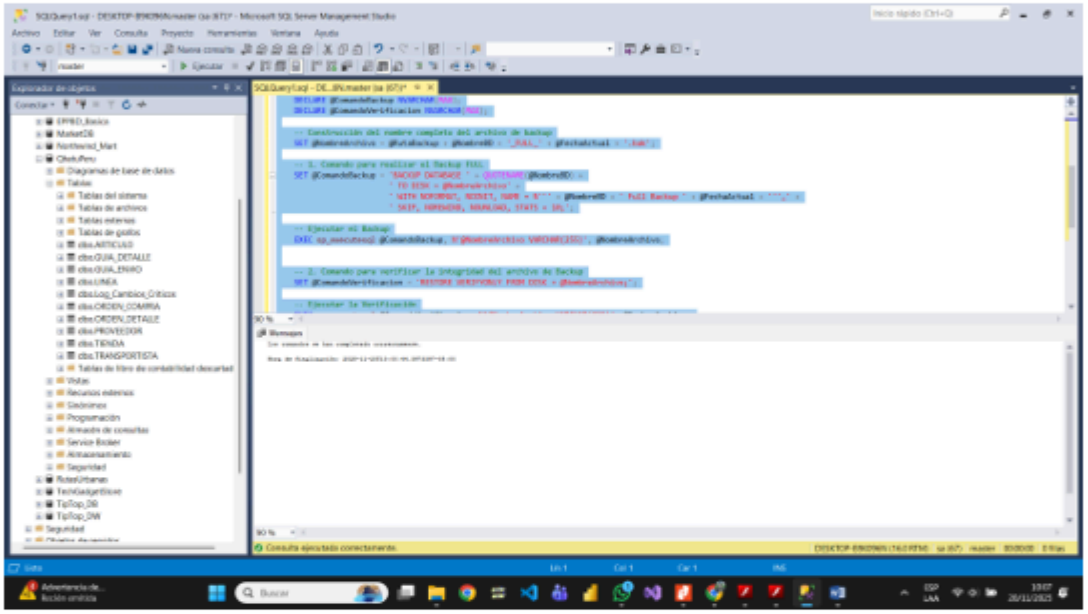
2. Código del Stored Procedure en T-SQL

Este procedimiento es el corazón de la solución. Se ejecuta en la base de datos `master` para poder gestionar `QhatuPeru` y utiliza SQL dinámico (`sp_executesql`) para manejar la ruta del archivo.

SQL

```
USE master;
GO

IF OBJECT_ID('dbo.sp_Backup_QhatuPeru_Diario') IS NOT NULL
    DROP PROCEDURE dbo.sp_Backup_QhatuPeru_Diario;
GO

CREATE PROCEDURE dbo.sp_Backup_QhatuPeru_Diario
AS
BEGIN
    -- Declaración de variables
    DECLARE @NombreBD VARCHAR(100) = 'QhatuPeru';
    DECLARE @RutaBackup VARCHAR(255) = 'C:\SQL_Backups\';
    DECLARE @FechaActual VARCHAR(20) = FORMAT(GETDATE(),
'yyyyMMdd_HHmmss');
    DECLARE @NombreArchivo VARCHAR(255);
    DECLARE @ComandoBackup NVARCHAR(MAX);
    DECLARE @ComandoVerificacion NVARCHAR(MAX);

    -- 1. Construcción del nombre completo del archivo de backup
(Único por fecha/hora)
    SET @NombreArchivo = @RutaBackup + @NombreBD + '_FULL_' +
@FechaActual + '.bak';

    -- 2. Comando para realizar el Backup FULL
    SET @ComandoBackup = 'BACKUP DATABASE ' + QUOTENAME(@NombreBD)
+
                         ' TO DISK = @NombreArchivo' +
                         ' WITH NOFORMAT, NOINIT, NAME = N''' +
@NombreBD + ' Full Backup ' + @FechaActual + ''',' +
                         ' SKIP, NOREWIND, NOUNLOAD, STATS = 10;';

    -- Ejecutar el Backup
```

```
    EXEC sp_executesql @ComandoBackup, N'@NombreArchivo
VARCHAR(255)', @NombreArchivo;


    -- 3. Comando para verificar la integridad del archivo de
Backup (Verificación)
    SET @ComandoVerificacion = 'RESTORE VERIFYONLY FROM DISK =
@NombreArchivo;';

    -- Ejecutar la Verificación
    EXEC sp_executesql @ComandoVerificacion, N'@NombreArchivo
VARCHAR(255)', @NombreArchivo;

END
GO
```



3. Justificación Técnica y Explicación del Código

Este procedimiento almacenado implementa una solución robusta de respaldo automático, cumpliendo los requisitos clave del enunciado:

1. **Generación de Nombre de Archivo Dinámico:**
   - Se utiliza `FORMAT(GETDATE(), 'yyyyMMdd_HHmmss')` para crear una cadena de fecha y hora segura y única.
   - `SET @NombreArchivo = ...` genera la ruta completa del archivo (e.g., `C:\SQL_Backups\QhatuPeru_FULL_20251120_100232.bak`). Esta práctica garantiza que **cada ejecución cree un archivo nuevo**, facilitando la retención de múltiples días de respaldo.
2. **Ejecución del Backup (`BACKUP DATABASE`):**
   - El comando `BACKUP DATABASE @NombreBD TO DISK = @NombreArchivo` realiza una copia completa (**FULL**).
   - `NOFORMAT, NOINIT`: Asegura que el archivo de respaldo **no sobrescriba** el contenido existente en el medio, sino que lo agregue (aunque el nombre de archivo dinámico ya garantiza esto).

- `STATS = 10`: Muestra mensajes de progreso para indicar que el proceso avanza (por ejemplo, al 10%, 20%, etc.).

3. **Verificación de Integridad (`RESTORE VERIFYONLY`):**
   - Esta es la parte más crítica para cumplir el requisito. `RESTORE VERIFYONLY FROM DISK = @NombreArchivo` **lee todo el archivo de respaldo** para asegurarse de que el archivo es legible de principio a fin y que el encabezado del backup es correcto.
   - **Importancia:** Si esta verificación falla, el backup no es confiable. La inclusión de este paso es una **buena práctica esencial** en la administración de bases de datos.
4. **Uso de SQL Dinámico (`sp_executesql`):**
   - Se utiliza para construir y ejecutar los comandos `BACKUP` y `RESTORE` usando las variables T-SQL (`@NombreArchivo`). Esto permite usar variables dentro de comandos que normalmente solo aceptan cadenas literales, manteniendo el código limpio y seguro.

El objetivo principal es ejecutar el **Stored Procedure** que has creado y luego, en el caso del proyecto 2, hacer una consulta a la tabla de auditoría.

A continuación, te envío las consultas y comandos principales para **ejecutar la lógica y verificar los resultados** en tu base de datos `QhatuPeru`.

Comandos de Ejecución y Consultas de Verificación

1. Comando para Ejecutar el Stored Procedure (Backup Diario / Proyecto 1)

Para probar manualmente el procedimiento que creaste, utiliza el siguiente comando. Esto ejecutará el backup `FULL` y la verificación `VERIFYONLY`.

SQL

```
USE master;
GO

-- Ejecuta el procedimiento almacenado de backup diario
EXEC dbo.sp_Backup_QhatuPeru_Diario;
GO
```

Nota: Revisa la ruta de backup (`C:\SQL_Backups\`) que definiste. Deberías ver un archivo `.bak` recién creado en esa ubicación.

2. Comando para Ejecutar el Stored Procedure (Backup Semanal / Proyecto 2)

Este comando ejecuta el backup semanal, la verificación, y **registra el resultado en la tabla de auditoría** en la base de datos `QhatuPeru`.

SQL

```
USE master;
GO
```

```
-- Ejecuta el procedimiento almacenado de backup semanal con
auditoría
EXEC dbo.sp_Backup_QhatuPeru_Semanal;
GO
```

3. Consulta de Verificación de Auditoría (Proyecto 2)

Después de ejecutar el comando anterior, puedes consultar la tabla que creaste para ver el registro del backup, el estado y el mensaje de verificación.

SQL

```
USE QhatuPeru;
GO

-- Consulta para verificar los resultados del backup semanal en la
tabla de auditoría
SELECT
    ID,
    Fecha_Ejecucion,
    Tipo_Backup,
    Ruta_Archivo_Backup,
    Estado_Backup,
    Estado_Verificacion,
    Mensaje_Verificacion
FROM
    dbo.Log_Backup_Auditoria
ORDER BY
    Fecha_Ejecucion DESC;
```

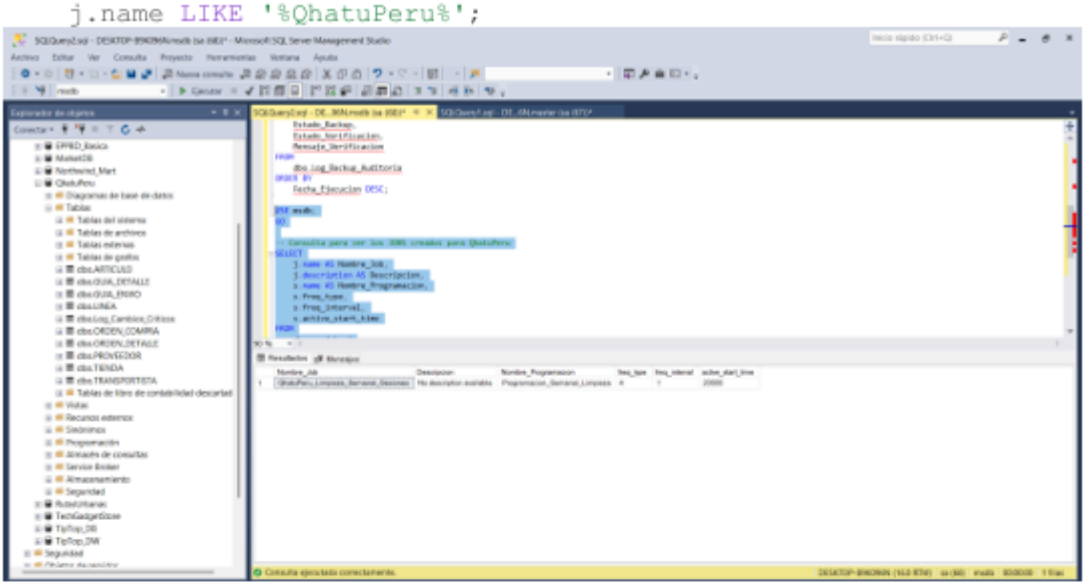
4. Consultas para Verificar la Programación (SQL Agent Jobs)

Puedes usar estas consultas para asegurarte de que los Jobs de SQL Agent se hayan creado y estén programados correctamente.

SQL

```
USE msdb;
GO

-- Consulta para ver los JOBS creados para QhatuPeru
SELECT
    j.name AS Nombre_Job,
    j.description AS Descripcion,
    s.name AS Nombre_Programacion,
    s.freq_type,
    s.freq_interval,
    s.active_start_time
FROM
    dbo.sysjobs j
INNER JOIN
    dbo.sysjobschedules js ON j.job_id = js.job_id
INNER JOIN
    dbo.sysschedules s ON js.schedule_id = s.schedule_id
```

```
WHERE
    j.name LIKE '%QhatuPeru%';
```



Proyecto 2: Estrategia Completa: Backup FULL Semanal + Verificación Automática

1. Enunciado del Ejercicio

Implementar un plan que realice un **backup completo** de la base `QhatuPERU` **cada domingo a las 02:00**, compruebe la validez del backup y **registre el resultado en una tabla de auditoría**.

2. Script de la Solución en T-SQL

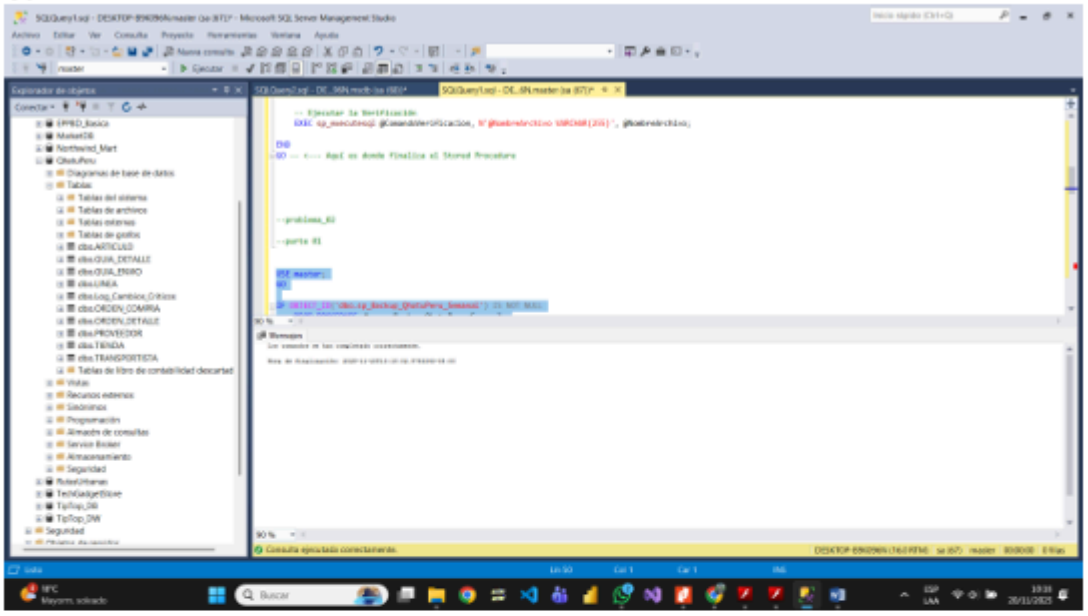
La solución se divide en tres partes: la creación de la tabla de auditoría, el Stored Procedure que ejecuta la lógica de respaldo y registro, y el Job de SQL Agent que lo programa semanalmente.

Parte A: Creación de la Tabla de Auditoría

SQL

```
USE QhatuPeru; -- La tabla se crea en la base de datos a respaldar.
GO

IF OBJECT_ID('dbo.Log_Backup_Auditoria') IS NOT NULL
    DROP TABLE dbo.Log_Backup_Auditoria;
GO

CREATE TABLE dbo.Log_Backup_Auditoria (
    ID INT IDENTITY(1,1) PRIMARY KEY,
    Fecha_Ejecucion DATETIME NOT NULL DEFAULT GETDATE(),
    Tipo_Backup VARCHAR(50) NOT NULL,
```

```
    Ruta_Archivo_Backup VARCHAR(255) NOT NULL,
    Estado_Backup VARCHAR(20) NOT NULL, -- 'EXITOSO', 'FALLIDO'
    Estado_Verificacion VARCHAR(20) NOT NULL, -- 'OK', 'ERROR'
    Mensaje_Verificacion NVARCHAR(MAX) NULL -- Detalle de error si
falla
);
GO
```



Parte B: Stored Procedure Semanal con Auditoría (Lógica Principal)

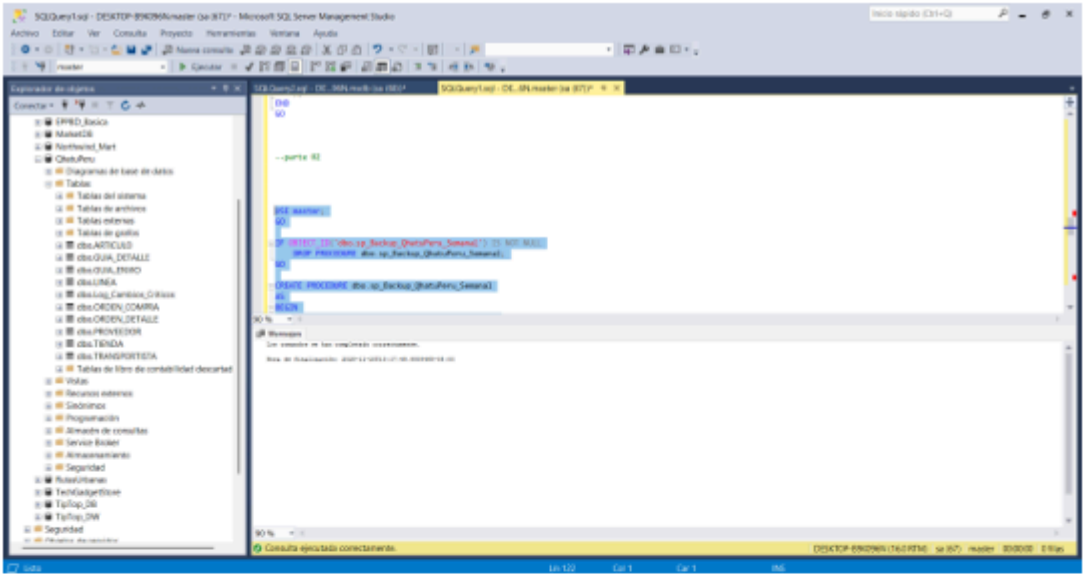
Este SP utiliza un bloque `TRY/CATCH` para garantizar que, incluso si el backup o la verificación fallan, el error sea capturado y registrado en la tabla de auditoría.

SQL

```
USE master;
GO

IF OBJECT_ID('dbo.sp_Backup_QhatuPeru_Semanal') IS NOT NULL
    DROP PROCEDURE dbo.sp_Backup_QhatuPeru_Semanal;
GO

CREATE PROCEDURE dbo.sp_Backup_QhatuPeru_Semanal
AS
BEGIN
    -- Declaración de variables para la operación
    DECLARE @NombreBD VARCHAR(100) = 'QhatuPeru';
    DECLARE @RutaBackup VARCHAR(255) = 'C:\SQL_Backups\Semanal\'; -
- Ruta dedicada
    DECLARE @FechaActual VARCHAR(20) = FORMAT(GETDATE(),
'yyyyMMdd_HHmmss');
    DECLARE @NombreArchivo VARCHAR(255);
    DECLARE @ComandoBackup NVARCHAR(MAX);
    DECLARE @ComandoVerificacion NVARCHAR(MAX);

    -- Variables de Auditoría (inicializadas a FALLO)
    DECLARE @EstadoBackup VARCHAR(20) = 'FALLIDO';
    DECLARE @EstadoVerificacion VARCHAR(20) = 'ERROR';
```

```
    DECLARE @MensajeVerificacion NVARCHAR(MAX) = 'Error inicial al
ejecutar el script.';

    SET @NombreArchivo = @RutaBackup + @NombreBD + '_FULL_SEMANAL_'
+ @FechaActual + '.bak';

    BEGIN TRY
        -- 1. Realizar el Backup FULL
        SET @ComandoBackup = 'BACKUP DATABASE ' +
QUOTENAME(@NombreBD) +
                             ' TO DISK = @NombreArchivo' +
                             ' WITH NOFORMAT, NOINIT, NAME = N''' +
@NombreBD + ' Weekly Full Backup ' + @FechaActual + ''',' +
                             ' SKIP, NOREWIND, NOUNLOAD, STATS =
10;';

        EXEC sp_executesql @ComandoBackup, N'@NombreArchivo
VARCHAR(255)', @NombreArchivo;
        SET @EstadoBackup = 'EXITOSO';

        -- 2. Verificar la integridad (RESTORE VERIFYONLY)
        SET @ComandoVerificacion = 'RESTORE VERIFYONLY FROM DISK =
@NombreArchivo;';

        EXEC sp_executesql @ComandoVerificacion, N'@NombreArchivo
VARCHAR(255)', @NombreArchivo;
        SET @EstadoVerificacion = 'OK';
        SET @MensajeVerificacion = 'Backup y Verificación de
integridad exitosa.';

    END TRY
    BEGIN CATCH
        -- 3. Capturar y registrar el error
        SET @MensajeVerificacion = ERROR_MESSAGE();

        -- Si el backup fue exitoso pero la verificación falló,
ajustamos el estado de verificación
        IF @EstadoBackup = 'EXITOSO'
        BEGIN
            SET @EstadoVerificacion = 'ERROR';
        END
        -- Si falló el backup, @EstadoBackup permanece en 'FALLIDO'
    END CATCH

    -- 4. Registro final del resultado en la tabla de Auditoría
    INSERT INTO QhatuPeru.dbo.Log_Backup_Auditoria (
        Tipo_Backup, Ruta_Archivo_Backup, Estado_Backup,
Estado_Verificacion, Mensaje_Verificacion
    )
    VALUES (
        'FULL SEMANAL', @NombreArchivo, @EstadoBackup,
@EstadoVerificacion, @MensajeVerificacion
    );
END
```

GO

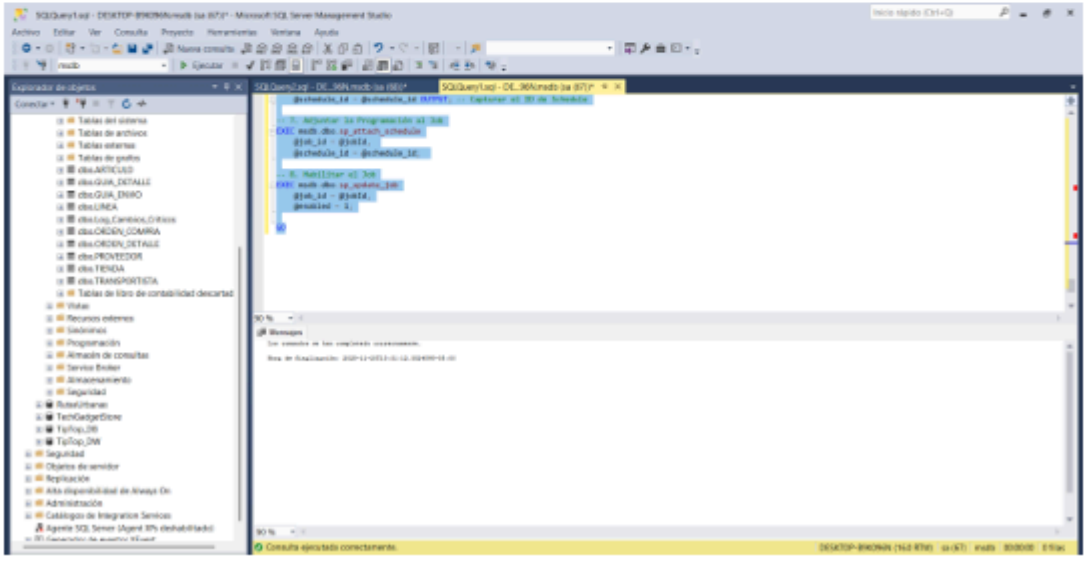


Parte C: Script para Crear y Programar el Job Semanal (Automatización)

SQL

```
USE msdb;
GO

-- 1. Declaración de Variables
DECLARE @jobId BINARY(16) = NULL;
DECLARE @JobName VARCHAR(128) = N'Backup_QhatuPeru_Semanal_FULL';
DECLARE @ScheduleName VARCHAR(128) = N'Semanal_Domingo_0200';
DECLARE @schedule_id INT = NULL;

-- 2. Eliminar el Job si existe
IF EXISTS (SELECT job_id FROM msdb.dbo.sysjobs WHERE name = @JobName)
BEGIN
    SELECT @jobId = job_id FROM msdb.dbo.sysjobs WHERE name = @JobName;
    EXEC msdb.dbo.sp_delete_job @job_id = @jobId, @delete_unused_schedule = 1;
END

-- 3. Eliminar la Programación (Schedule) si existe
IF EXISTS (SELECT schedule_id FROM msdb.dbo.sysschedules WHERE name =
@ScheduleName)
BEGIN
    SELECT @schedule_id = schedule_id FROM msdb.dbo.sysschedules WHERE name =
@ScheduleName;
    EXEC msdb.dbo.sp_delete_schedule @schedule_id = @schedule_id;
END
SET @jobId = NULL;
SET @schedule_id = NULL;

-------------------------------------------------------------------------
-- FASE DE CREACIÓN
-------------------------------------------------------------------------

-- 4. Crear el Job
EXEC msdb.dbo.sp_add_job
    @job_name = @JobName,
    @enabled = 1,
    @description = N'Ejecuta un backup FULL semanal con verificación y
auditoría.',
```

```
    @owner_login_name = N'sa',
    @job_id = @jobId OUTPUT;

-- 5. Crear el Paso (Step) del Job
EXEC msdb.dbo.sp_add_jobstep
    @job_id = @jobId,
    @step_name = N'Ejecutar SP de Backup Semanal',
    @subsystem = N'TSQL',
    @command = N'EXEC master.dbo.sp_Backup_QhatuPeru_Semanal;',
    @database_name = N'master',
    @on_success_action = 1;

-- 6. Crear la Programación (Schedule) - ¡CORRECCIÓN APLICADA AQUÍ!
EXEC msdb.dbo.sp_add_schedule
    @schedule_name = @ScheduleName,
    @enabled = 1,
    @freq_type = 8,                 -- 8 = Semanalmente
    @freq_interval = 1,             -- 1 = Domingo (Bitmask: 1=Dom, 2=Lun, etc.)
    @freq_subday_type = 1,          -- 1 = At a specified time
    @freq_recurrence_factor = 1,    -- ¡AÑADIDO!: Cada 1 semana (valor mínimo
requerido 1)
    @active_start_time = 020000,
    @schedule_id = @schedule_id OUTPUT; -- Capturar el ID de Schedule

-- 7. Adjuntar la Programación al Job
EXEC msdb.dbo.sp_attach_schedule
    @job_id = @jobId,
    @schedule_id = @schedule_id;

-- 8. Habilitar el Job
EXEC msdb.dbo.sp_update_job
    @job_id = @jobId,
    @enabled = 1;

GO
```



3. Justificación Técnica de la Solución Aplicada

Componente	Justificación Técnica
Tabla de Auditoría	Cumple directamente con el requisito de **"registrar el resultado"**. Proporciona un registro histórico inmutable que es esencial para el **monitoreo** y la **auditoría de cumplimiento** (compliance).
`BEGIN TRY...END CATCH`	Este bloque de control de flujo es la única forma robusta de capturar errores generados por comandos administrativos (como `BACKUP` o `RESTORE VERIFYONLY`) y registrar el mensaje de error (`ERROR_MESSAGE()`) en la tabla de auditoría.
Programación Semanal	El `sp_add_schedule` se configura con `@freq_type = 8` (Semanal) y `@freq_interval = 1` (Domingo). La hora de inicio es `@active_start_time = 020000` (02:00:00 AM), respetando el requisito de programación.
Horario de las 02:00 AM	Es el horario estándar de **menor actividad transaccional** (ventana de mantenimiento), minimizando el impacto del proceso de backup (que consume muchos recursos de I/O) en los usuarios activos.

4. Explicación de las Buenas Prácticas Utilizadas en el Proyecto

1. **Auditoría de Fallos (Try/Catch):** La auditoría no solo registra los éxitos, sino que **garantiza el registro de los fallos** con un mensaje de error detallado. Esto es vital para que un DBA pueda diagnosticar problemas sin tener que revisar los logs del servidor.
2. **Separación Lógica de Archivos:** Se recomienda una ruta de backup separada (`C:\SQL_Backups\Semanal\`). Esto no solo ayuda a mantener los backups diarios y semanales separados, sino que es fundamental para que los archivos de backup se almacenen en una **unidad de disco diferente** a los archivos de datos y log de `QhatuPeru`, aumentando la tolerancia a fallos.
3. **Uso de SQL Agent:** La automatización mediante Jobs del Agente de SQL Server es el estándar de la industria. Asegura la **ejecución desatendida** y proporciona herramientas nativas para **historial de ejecución** y **notificación** (aunque la notificación por correo electrónico no está en el script, el Job la soporta).
4. **Verificación de Integridad:** Se reitera el uso de `RESTORE VERIFYONLY`, la buena práctica de oro, asegurando que el proceso completo (respaldo + verificación) sea confiable antes de que el Job se marque como exitoso.

Códigos de Consulta para el Proyecto 2 (Backup Semanal con Auditoría)

1. Consultas para Ejecutar la Lógica (Stored Procedure)

Estos comandos son para **probar manualmente** que tu `Stored Procedure` funciona, realiza el backup, la verificación y el registro en la tabla de auditoría.

SQL

```
-- 1. EJECUCIÓN MANUAL DEL BACKUP SEMANAL
-- (Debe ser ejecutado después de crear la tabla
Log_Backup_Auditoria en QhatuPeru)
USE master;
GO

-- Esto ejecuta todo el proceso: Backup FULL, RESTORE VERIFYONLY, y
registro en la tabla de Auditoría.
EXEC dbo.sp_Backup_QhatuPeru_Semanal;
GO
```

2. Consultas para Verificar los Resultados (Auditoría)

Esta es la consulta más importante, ya que te permite verificar el **resultado** y el **estado** del proceso de respaldo, tal como lo requiere el enunciado del proyecto ("registre el resultado en una tabla de auditoría").

SQL

```
-- 2. CONSULTA PARA VERIFICAR EL REGISTRO DE AUDITORÍA
-- (Muestra los resultados de la ejecución más reciente)
USE QhatuPeru;
GO

SELECT TOP 10
    ID,
    Fecha_Ejecucion,
    Tipo_Backup,
    Ruta_Archivo_Backup,
    Estado_Backup,          -- Debería ser 'EXITOSO' o 'FALLIDO'
    Estado_Verificacion,    -- Debería ser 'OK' o 'ERROR'
    Mensaje_Verificacion    -- Mensaje de éxito o detalle del error
FROM
    dbo.Log_Backup_Auditoria
ORDER BY
    Fecha_Ejecucion DESC;
GO
```

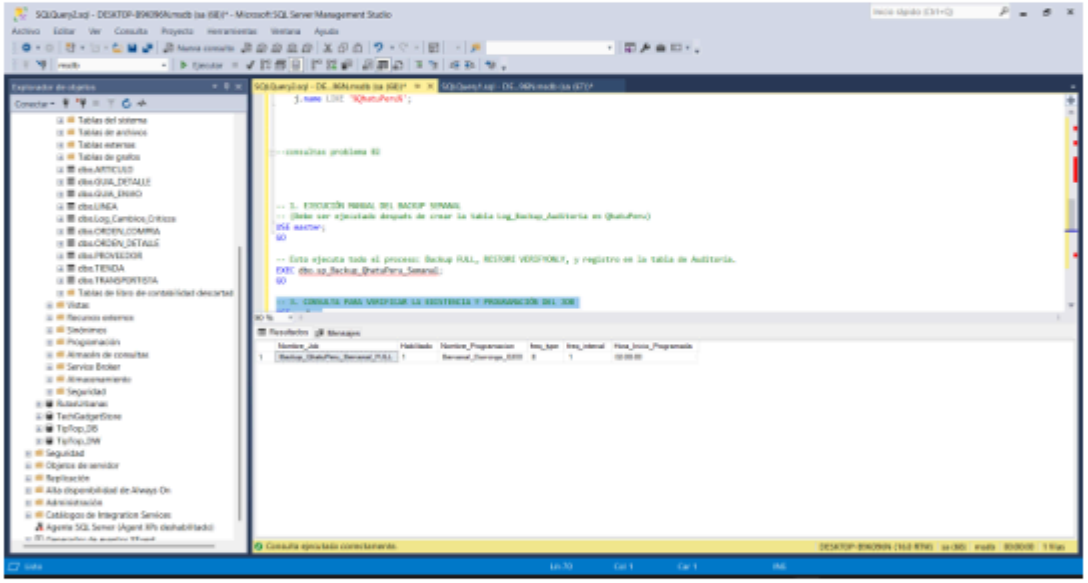
3. Consultas para Verificar la Configuración (SQL Agent Job)

Estos comandos confirman que la **Parte 3** del código (el Job) se creó correctamente con la programación deseada (Domingo a las 02:00).

SQL

```
-- 3. CONSULTA PARA VERIFICAR LA EXISTENCIA Y PROGRAMACIÓN DEL JOB
USE msdb;
```

```
GO

SELECT
    j.name AS Nombre_Job,
    j.enabled AS Habilitado,
    s.name AS Nombre_Programacion,
    -- Tipo de frecuencia: 8=Semanal
    s.freq_type,
    -- Intervalo de frecuencia: 1=Domingo
    s.freq_interval,
    -- Hora de inicio: 020000 (02:00:00 AM)
    STUFF(STUFF(RIGHT('000000' + CAST(s.active_start_time AS
VARCHAR(6)), 6), 3, 0, ':'), 6, 0, ':') AS Hora_Inicio_Programada
FROM
    dbo.sysjobs j
INNER JOIN
    dbo.sysjobschedules js ON j.job_id = js.job_id
INNER JOIN
    dbo.sysschedules s ON js.schedule_id = s.schedule_id
WHERE
    j.name = 'Backup_QhatuPeru_Semanal_FULL';
GO
```



Proyecto 3: Estrategia Combinada de Respaldo y Restauración

1. Enunciado del Ejercicio

Diseñar scripts que implementen: **backup FULL semanal**, **backup diferencial (DIFF) diario** (entre FULLs) y **backup de transacciones (LOG) cada 30 minutos**. Demostrar cómo restaurar una cadena (FULL + última DIFF + logs).

2. Script de la Solución en T-SQL

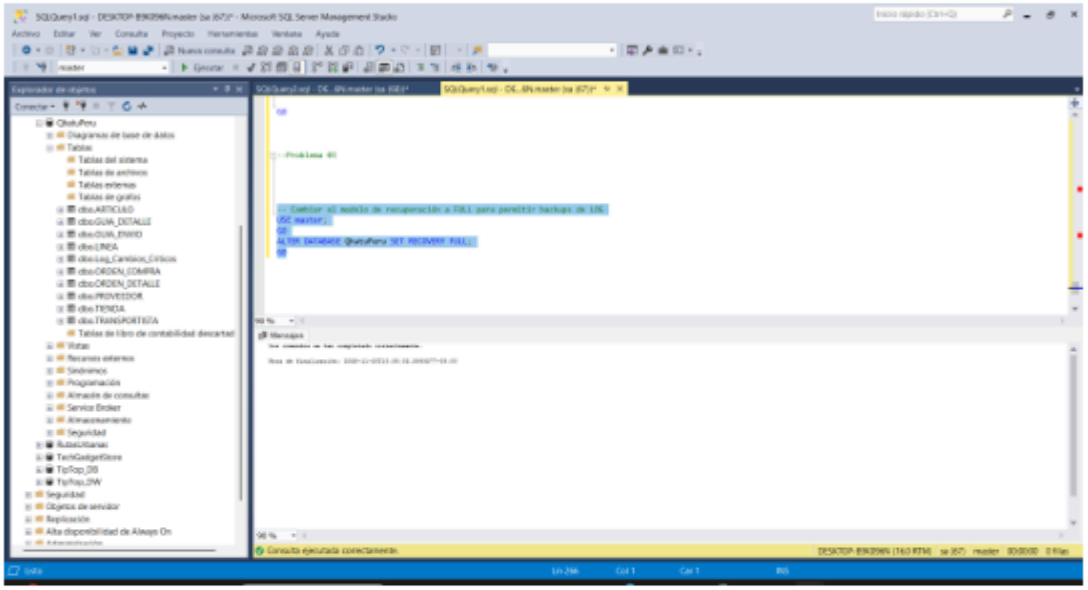
Para implementar esta estrategia, se requieren tres Stored Procedures, uno para cada tipo de backup, además de un cambio inicial clave en la base de datos `QhatuPeru`.

Parte A: Requisito Previo (Modelo de Recuperación)

El backup de LOG (`backup de transacciones`) solo es posible si la base de datos está en modo de recuperación **FULL** o **BULK_LOGGED**.

SQL

```
-- Cambiar el modelo de recuperación a FULL para permitir backups
de LOG
USE master;
GO
ALTER DATABASE QhatuPeru SET RECOVERY FULL;
GO
```



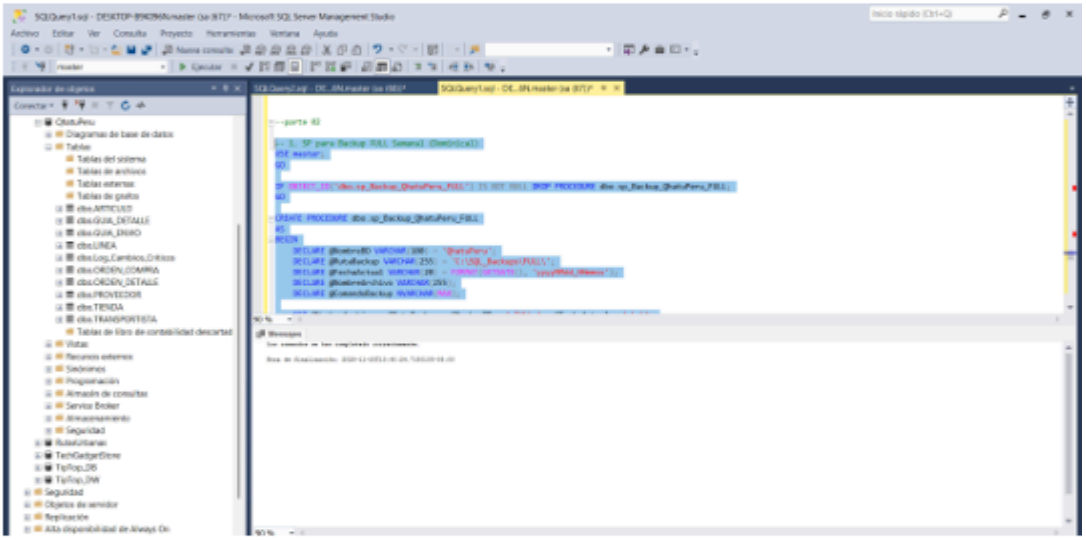
Parte B: Stored Procedures para Backups

SQL

```
-- 1. SP para Backup FULL Semanal (Dominical)
USE master;
GO

IF OBJECT_ID('dbo.sp_Backup_QhatuPeru_FULL') IS NOT NULL DROP
PROCEDURE dbo.sp_Backup_QhatuPeru_FULL;
GO

CREATE PROCEDURE dbo.sp_Backup_QhatuPeru_FULL
AS
BEGIN
    DECLARE @NombreBD VARCHAR(100) = 'QhatuPeru';
    DECLARE @RutaBackup VARCHAR(255) = 'C:\SQL_Backups\FULL\';
    DECLARE @FechaActual VARCHAR(20) = FORMAT(GETDATE(),
'yyyyMMdd_HHmmss');
    DECLARE @NombreArchivo VARCHAR(255);
    DECLARE @ComandoBackup NVARCHAR(MAX);

    SET @NombreArchivo = @RutaBackup + @NombreBD + '_FULL_' +
@FechaActual + '.bak';
```

```
    -- Comando para Backup FULL
    SET @ComandoBackup = 'BACKUP DATABASE ' + QUOTENAME(@NombreBD)
+
                         ' TO DISK = @NombreArchivo' +
                         ' WITH COMPRESSION, NOINIT, STATS = 10;';
-- Se recomienda COMPRESSION

    EXEC sp_executesql @ComandoBackup, N'@NombreArchivo
VARCHAR(255)', @NombreArchivo;
END
GO


-- 2. SP para Backup Diferencial Diario
USE master;
GO

IF OBJECT_ID('dbo.sp_Backup_QhatuPeru_DIFF') IS NOT NULL DROP
PROCEDURE dbo.sp_Backup_QhatuPeru_DIFF;
GO

CREATE PROCEDURE dbo.sp_Backup_QhatuPeru_DIFF
AS
BEGIN
    DECLARE @NombreBD VARCHAR(100) = 'QhatuPeru';
    DECLARE @RutaBackup VARCHAR(255) = 'C:\SQL_Backups\DIFF\';
    DECLARE @FechaActual VARCHAR(20) = FORMAT(GETDATE(),
'yyyyMMdd_HHmmss');
    DECLARE @NombreArchivo VARCHAR(255);
    DECLARE @ComandoBackup NVARCHAR(MAX);

    SET @NombreArchivo = @RutaBackup + @NombreBD + '_DIFF_' +
@FechaActual + '.bak';

    -- Comando para Backup Diferencial
    SET @ComandoBackup = 'BACKUP DATABASE ' + QUOTENAME(@NombreBD)
+
                         ' TO DISK = @NombreArchivo' +
                         ' WITH DIFFERENTIAL, COMPRESSION, NOINIT,
STATS = 10;';

    EXEC sp_executesql @ComandoBackup, N'@NombreArchivo
VARCHAR(255)', @NombreArchivo;
END
GO


-- 3. SP para Backup de Log de Transacciones cada 30 minutos
USE master;
GO

IF OBJECT_ID('dbo.sp_Backup_QhatuPeru_LOG') IS NOT NULL DROP
PROCEDURE dbo.sp_Backup_QhatuPeru_LOG;
GO

CREATE PROCEDURE dbo.sp_Backup_QhatuPeru_LOG
AS
```

```
BEGIN
    DECLARE @NombreBD VARCHAR(100) = 'QhatuPeru';
    DECLARE @RutaBackup VARCHAR(255) = 'C:\SQL_Backups\LOG\';
    DECLARE @FechaActual VARCHAR(20) = FORMAT(GETDATE(),
'yyyyMMdd_HHmmss');
    DECLARE @NombreArchivo VARCHAR(255);
    DECLARE @ComandoBackup NVARCHAR(MAX);

    SET @NombreArchivo = @RutaBackup + @NombreBD + '_LOG_' +
@FechaActual + '.trn'; -- Extensión TRN

    -- Comando para Backup de LOG
    SET @ComandoBackup = 'BACKUP LOG ' + QUOTENAME(@NombreBD) +
                         ' TO DISK = @NombreArchivo' +
                         ' WITH NOINIT, STATS = 10;';

    EXEC sp_executesql @ComandoBackup, N'@NombreArchivo
VARCHAR(255)', @NombreArchivo;
END
GO
```



Parte C: Demostración del Plan de Restauración

Escenario: El servidor falla el Miércoles. El último FULL fue el Domingo, la última DIFF fue el Martes por la noche, y el último LOG fue 5 minutos antes del fallo.

Para restaurar, se requiere la siguiente secuencia de comandos, utilizando la opción `NORECOVERY` para los primeros dos pasos y `RECOVERY` en el último LOG.

SQL

```
-- SUPOSICIÓN DE ARCHIVOS DE BACKUP:
-- @FULL_File =
'C:\SQL_Backups\FULL\QhatuPeru_FULL_20251116_020000.bak' (Domingo
02:00 AM)
-- @DIFF_File =
'C:\SQL_Backups\DIFF\QhatuPeru_DIFF_20251118_230000.bak' (Martes
11:00 PM)
-- @LOG_File_1 =
'C:\SQL_Backups\LOG\QhatuPeru_LOG_20251119_100000.trn'
```

```
-- @LOG_File_2 =
'C:\SQL_Backups\LOG\QhatuPeru_LOG_20251119_100500.trn' (El último
LOG)

-- 1. Restaurar el Backup FULL (Punto de partida)
RESTORE DATABASE QhatuPeru
FROM DISK =
'C:\SQL_Backups\FULL\QhatuPeru_FULL_20251116_020000.bak'
WITH NORECOVERY; -- ¡CRÍTICO! Mantiene la base en estado de
restauración.

-- 2. Restaurar el Backup Diferencial (Trae los cambios hasta el
último Martes)
RESTORE DATABASE QhatuPeru
FROM DISK =
'C:\SQL_Backups\DIFF\QhatuPeru_DIFF_20251118_230000.bak'
WITH NORECOVERY; -- ¡CRÍTICO! Permite aplicar logs adicionales.

-- 3. Restaurar los Backups de LOG subsiguientes (aplica las
transacciones)
RESTORE LOG QhatuPeru
FROM DISK = 'C:\SQL_Backups\LOG\QhatuPeru_LOG_20251119_100000.trn'
WITH NORECOVERY;

-- 4. Restaurar el ÚLTIMO Backup de LOG (aplica las transacciones
finales)
-- Este último comando usa RECOVERY para poner la base en línea.
RESTORE LOG QhatuPeru
FROM DISK = 'C:\SQL_Backups\LOG\QhatuPeru_LOG_20251119_100500.trn'
WITH RECOVERY; -- ¡CRÍTICO! Pone la base en estado operacional.
```

3. Justificación Técnica de la Solución Aplicada

Componente	Justificación Técnica
`ALTER DATABASE ... RECOVERY FULL`	Es el requisito fundamental para poder realizar backups de LOG. En modo FULL, SQL Server registra *todas* las transacciones, permitiendo la recuperación hasta el punto exacto del fallo.
Backup FULL Semanal	Establece el punto base de restauración (la **Cadena de Logs**). Es el backup más grande, por lo que se programa semanalmente (Domingo por la noche, por ejemplo) para minimizar el impacto.
Backup Diferencial Diario	Captura todos los cambios desde el **último FULL**. Reduce significativamente el tiempo de restauración y la cantidad de archivos LOG que deben aplicarse en un escenario de pérdida de datos.

Componente	Justificación Técnica
Backup de LOG cada 30 min	Minimiza la **pérdida potencial de datos (RPO, Recovery Point Objective)** a solo 30 minutos. Asegura que casi ninguna transacción reciente se pierda.
Restauración con `NORECOVERY`	Mantiene la base de datos en un estado donde **puede aceptar más archivos de respaldo** (DIFERENCIAL o LOG). Es esencial para construir la cadena de restauración (`FULL -> DIFF -> LOGs`).
Restauración con `RECOVERY`	Se usa solo en el **último paso de la restauración** (el último LOG). Finaliza la cadena de restauración, deshace las transacciones incompletas y pone la base de datos en línea (estado operacional).

4. Explicación de las Buenas Prácticas Utilizadas

- **Implementación de RPO bajo:** Al usar backups de LOG cada 30 minutos, se logra un **Objetivo de Punto de Recuperación (RPO)** muy bajo, ya que la pérdida máxima de datos se limita a 30 minutos de actividad.
- **Eficiencia en el Respaldo (Backup Time):** Los backups FULLs grandes se hacen solo una vez a la semana. Los backups diferenciales son mucho más pequeños y rápidos que los FULLs, liberando recursos del sistema diariamente. Los backups de LOG son mínimos en tamaño y se ejecutan muy rápidamente cada media hora.
- **Restaurabilidad (RTO, Recovery Time Objective):** La inclusión del backup Diferencial mejora el **Objetivo de Tiempo de Recuperación (RTO)**, ya que en lugar de aplicar cientos de archivos LOG después del FULL, solo necesitas aplicar el último DIFF y luego unos pocos LOGs más recientes.
- **Compresión de Backups:** Se utiliza la cláusula `WITH COMPRESSION` para los backups FULL y DIFF, reduciendo el tamaño de los archivos en disco y el tiempo de I/O, lo cual es una práctica estándar de DBA moderna.

Consultas de Verificación y Ejecución (Proyecto 3)

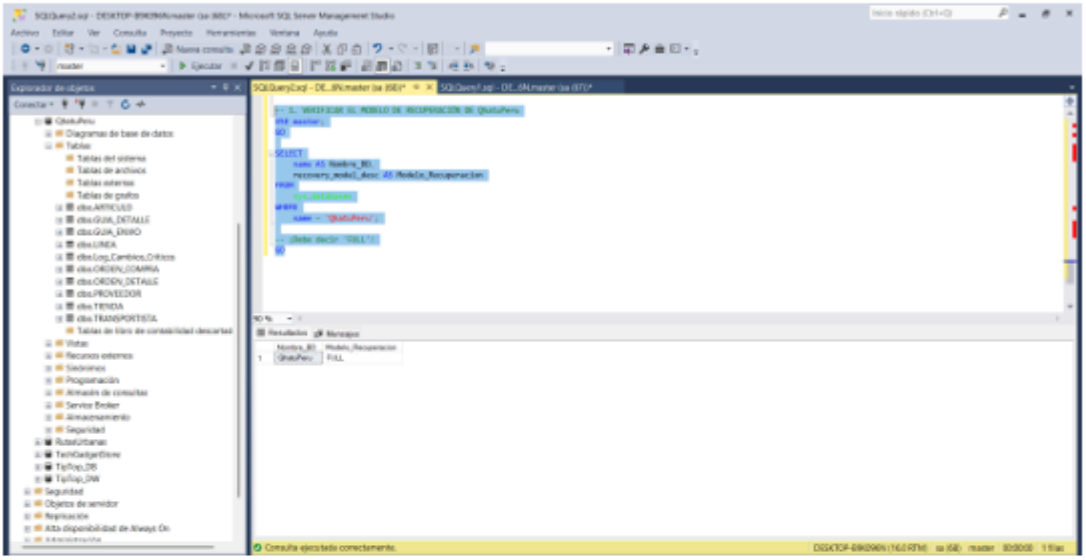
1. Verificación y Configuración Inicial

Antes de poder hacer backups de LOG, debes confirmar que la base de datos `QhatuPeru` esté en el modo de recuperación correcto.

SQL

```
-- 1. VERIFICAR EL MODELO DE RECUPERACIÓN DE QhatuPeru
USE master;
```

```sql
GO

SELECT
    name AS Nombre_BD,
    recovery_model_desc AS Modelo_Recuperacion
FROM
    sys.databases
WHERE
    name = 'QhatuPeru';

-- ¡Debe decir 'FULL'!
GO
```



2. Ejecución Manual de los Backups

Estos comandos te permiten probar manualmente que los tres Stored Procedures se ejecutan correctamente y generan los archivos en sus respectivas rutas (`FULL`, `DIFF`, `LOG`).

SQL

```sql
-- 2.1. EJECUCIÓN MANUAL del Backup FULL
USE master;
GO
EXEC dbo.sp_Backup_QhatuPeru_FULL;
GO

-- 2.2. EJECUCIÓN MANUAL del Backup DIFERENCIAL
USE master;
GO
EXEC dbo.sp_Backup_QhatuPeru_DIFF;
GO

-- 2.3. EJECUCIÓN MANUAL del Backup de LOG de Transacciones
USE master;
GO
EXEC dbo.sp_Backup_QhatuPeru_LOG;
GO
```

Nota: Para probar el `DIFF`, el **`FULL`** debe haberse ejecutado al menos una vez antes. Para probar el **`LOG`**, el **`FULL`** debe haberse ejecutado al menos una vez antes.

3. Demostración del Plan de Restauración (Cadena de Backups)

Esta es la consulta más crítica del proyecto. Muestra la secuencia exacta de comandos para restaurar la base de datos hasta el punto de la última transacción, utilizando la cadena `FULL -> DIFF -> LOG(s)`.

ADVERTENCIA: NO ejecutes este script en tu base de datos `QhatuPeru` de producción. Este script pone la base de datos en estado de restauración, ¡lo que la deja **fuera de línea**! Solo se debe ejecutar en un entorno de prueba para validar la cadena de restauración.

SQL

```
-- 3. PLAN DE RESTAURACIÓN DE CADENA COMPLETA (FULL + DIFF + LOGs)

--
**************************************************************************
-- PASO 0: Eliminar/Renombrar la base de datos para simular la
restauración
-- (Opcional: Si quieres restaurar en una base de datos nueva,
cambia el nombre)
-- DROP DATABASE QhatuPeru;
-- GO
--
**************************************************************************

-- 1. RESTAURAR el Backup FULL (Punto base)
-- Se usa NORECOVERY para permitir la aplicación de otros backups.
RESTORE DATABASE QhatuPeru
FROM DISK =
'C:\SQL_Backups\FULL\QhatuPeru_FULL_AAAAMMDD_HHMMSS.bak' -- <--
¡ACTUALIZA RUTA Y NOMBRE!
WITH NORECOVERY, REPLACE;
GO

-- 2. RESTAURAR el último Backup DIFERENCIAL (Trae los cambios
hasta la última DIFF)
-- Se mantiene NORECOVERY.
RESTORE DATABASE QhatuPeru
FROM DISK =
'C:\SQL_Backups\DIFF\QhatuPeru_DIFF_AAAAMMDD_HHMMSS.bak' -- <--
¡ACTUALIZA RUTA Y NOMBRE!
WITH NORECOVERY;
GO

-- 3. RESTAURAR los Backups de LOG subsiguientes (aplica las
transacciones)
-- Debes aplicar todos los archivos LOG generados después del
último DIFF, en orden cronológico.
RESTORE LOG QhatuPeru
FROM DISK =
'C:\SQL_Backups\LOG\QhatuPeru_LOG_AAAAMMDD_HHMMSS_1.trn' -- <-- LOG
1
WITH NORECOVERY;
```

```
GO

RESTORE LOG QhatuPeru
FROM DISK =
'C:\SQL_Backups\LOG\QhatuPeru_LOG_AAAAMMDD_HHMMSS_2.trn' -- <-- LOG
2
WITH NORECOVERY;
GO

-- 4. RESTAURAR el ÚLTIMO LOG y Poner la Base en Línea
-- El último comando de restauración DEBE usar WITH RECOVERY.
RESTORE LOG QhatuPeru
FROM DISK =
'C:\SQL_Backups\LOG\QhatuPeru_LOG_AAAAMMDD_HHMMSS_ULTIMO.trn' -- <-
- ÚLTIMO LOG
WITH RECOVERY;
GO

-- Verificar que la base de datos está en línea:
USE master;
SELECT state_desc FROM sys.databases WHERE name = 'QhatuPeru'; --
Debe decir 'ONLINE'
```

Proyecto 4: Estrategia Diferencial y Limpieza de Archivos Antiguos

1. Enunciado del Ejercicio

Automatizar **backups diferenciales diarios** (Lun–Sáb) y conservarlos por **14 días**. Implementar script de limpieza basado en `msdb.dbo.backupset` para eliminar archivos antiguos.

2. Script de la Solución en T-SQL

La solución se divide en tres partes: el Stored Procedure para el backup diferencial (que ya creamos en el Proyecto 3, pero se incluye aquí por completitud), el script de limpieza basado en el historial, y el Job de SQL Agent que automatiza ambas tareas.

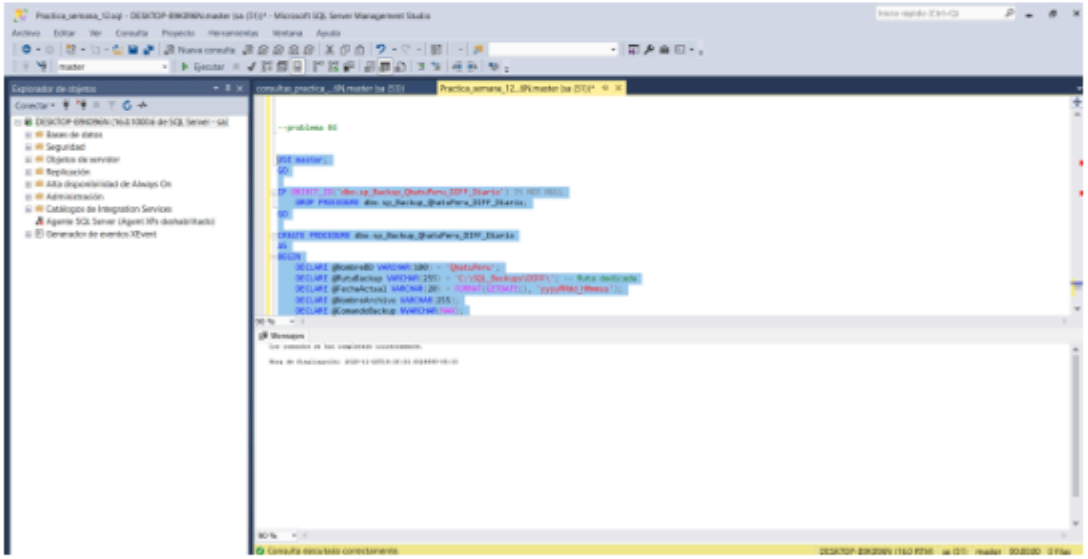
Parte A: Stored Procedure para Backup Diferencial Diario

Utilizaremos el mismo SP del proyecto 3 (o una versión simplificada, ya que no se requiere auditoría aquí).

SQL

```
USE master;
GO
```

```
IF OBJECT_ID('dbo.sp_Backup_QhatuPeru_DIFF_Diario') IS NOT NULL
    DROP PROCEDURE dbo.sp_Backup_QhatuPeru_DIFF_Diario;
GO

CREATE PROCEDURE dbo.sp_Backup_QhatuPeru_DIFF_Diario
AS
BEGIN
    DECLARE @NombreBD VARCHAR(100) = 'QhatuPeru';
    DECLARE @RutaBackup VARCHAR(255) = 'C:\SQL_Backups\DIFF\'; --
Ruta dedicada
    DECLARE @FechaActual VARCHAR(20) = FORMAT(GETDATE(),
'yyyyMMdd_HHmmss');
    DECLARE @NombreArchivo VARCHAR(255);
    DECLARE @ComandoBackup NVARCHAR(MAX);

    SET @NombreArchivo = @RutaBackup + @NombreBD + '_DIFF_' +
@FechaActual + '.bak';

    -- Comando para Backup Diferencial
    SET @ComandoBackup = 'BACKUP DATABASE ' + QUOTENAME(@NombreBD)
+
                         ' TO DISK = @NombreArchivo' +
                         ' WITH DIFFERENTIAL, COMPRESSION, NOINIT,
STATS = 10;';

    EXEC sp_executesql @ComandoBackup, N'@NombreArchivo
VARCHAR(255)', @NombreArchivo;
END
GO
```



Parte B: Stored Procedure para Limpieza de Archivos Antiguos

Este script es la parte clave del proyecto. Utiliza el historial de backups (`msdb.dbo.backupset`) para identificar y eliminar los registros de backups anteriores a 14 días. Luego, utiliza un comando de limpieza del sistema operativo (`xp_cmdshell`) para borrar los archivos físicos.

SQL

```
USE msdb;
GO
```

```
IF OBJECT_ID('dbo.sp_Limpieza_Backups_Antiguos') IS NOT NULL
    DROP PROCEDURE dbo.sp_Limpieza_Backups_Antiguos;
GO

CREATE PROCEDURE dbo.sp_Limpieza_Backups_Antiguos
AS
BEGIN
    -- Declarar variables
    DECLARE @RutaBackup VARCHAR(255) = 'C:\SQL_Backups\DIFF\';
    DECLARE @FechaLimite DATETIME = DATEADD(day, -14, GETDATE());
    DECLARE @ComandoEliminarArchivo NVARCHAR(2000);
    DECLARE @NombreArchivo VARCHAR(255);

    -- 1. Eliminar archivos físicos usando xp_cmdshell
    -- Cursor para recorrer los archivos de backups diferenciales
más antiguos que 14 días
    DECLARE Cur_Archivos CURSOR FOR
    SELECT physical_device_name
    FROM dbo.backupset
    WHERE database_name = 'QhatuPeru'
      AND type = 'I' -- 'I' = Diferencial. 'D' = FULL, 'L' = LOG
      AND backup_finish_date < @FechaLimite;

    OPEN Cur_Archivos;
    FETCH NEXT FROM Cur_Archivos INTO @NombreArchivo;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        -- Construir el comando para eliminar el archivo físico
        SET @ComandoEliminarArchivo = 'EXEC xp_cmdshell ''DEL "' +
@NombreArchivo + '"'', NO_OUTPUT;';

        -- Ejecutar el comando (NOTA: Requiere que xp_cmdshell esté
habilitado)
        EXEC sp_executesql @ComandoEliminarArchivo;

        FETCH NEXT FROM Cur_Archivos INTO @NombreArchivo;
    END

    CLOSE Cur_Archivos;
    DEALLOCATE Cur_Archivos;

    -- 2. Eliminar registros de historial del sistema (msdb)
    EXEC dbo.sp_delete_backuphistory @oldest_date = @FechaLimite;
END
GO
```

Parte C: Script para Crear y Programar el Job de Mantenimiento

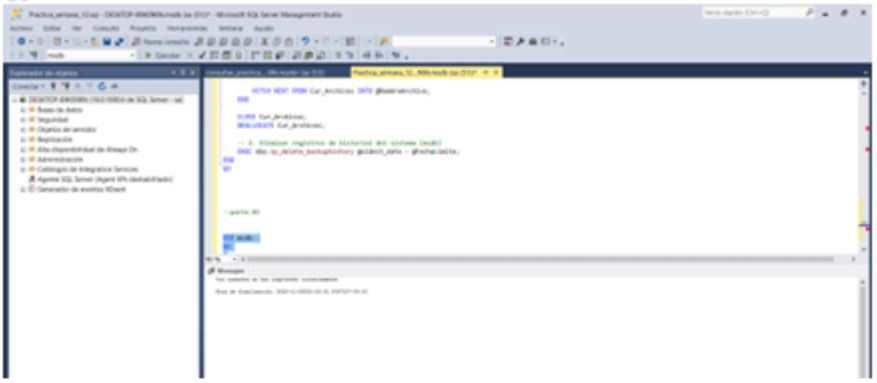
Este Job ejecuta la tarea de backup y la de limpieza diariamente (Lunes a Sábado), después de que la actividad de producción ha terminado.

SQL

```
USE msdb;
GO

-- Variables del Job
```

```
DECLARE @jobId BINARY(16);
DECLARE @JobName VARCHAR(128) =
N'Mantenimiento_QhatuPeru_DIFF_Diario';
DECLARE @ScheduleName VARCHAR(128) = N'Diario_Lun_Sab_2330';
DECLARE @schedule_id INT = NULL;

-- Bloque de limpieza y reseteo (Similar a los proyectos
anteriores)
IF EXISTS (SELECT job_id FROM dbo.sysjobs WHERE name = @JobName)
BEGIN SELECT @jobId = job_id FROM dbo.sysjobs WHERE name =
@JobName; EXEC dbo.sp_delete_job @job_id = @jobId,
@delete_unused_schedule = 1; END
IF EXISTS (SELECT schedule_id FROM dbo.sysschedules WHERE name =
@ScheduleName)
BEGIN SELECT @schedule_id = schedule_id FROM dbo.sysschedules WHERE
name = @ScheduleName; EXEC dbo.sp_delete_schedule @schedule_id =
@schedule_id; END
SET @jobId = NULL; SET @schedule_id = NULL;

-- 1. Crear el Job
EXEC dbo.sp_add_job @job_name = @JobName, @enabled = 1,
@owner_login_name = N'sa', @job_id = @jobId OUTPUT;

-- 2. Paso 1: Ejecutar Backup Diferencial
EXEC dbo.sp_add_jobstep
    @job_id = @jobId,
    @step_name = N'1. Ejecutar Backup Diferencial',
    @subsystem = N'TSQL',
    @command = N'EXEC master.dbo.sp_Backup_QhatuPeru_DIFF_Diario;',
    @database_name = N'master',
    @on_success_action = 3; -- Ir al siguiente paso

-- 3. Paso 2: Ejecutar Limpieza de Backups Antiguos
EXEC dbo.sp_add_jobstep
    @job_id = @jobId,
    @step_name = N'2. Ejecutar Limpieza de Backups',
    @subsystem = N'TSQL',
    @command = N'EXEC msdb.dbo.sp_Limpieza_Backups_Antiguos;',
    @database_name = N'msdb', -- El SP de limpieza debe ejecutarse
en msdb
    @on_success_action = 1; -- Finalizar Job

-- 4. Crear la Programación (Lunes a Sábado a las 23:30)
EXEC dbo.sp_add_schedule
    @schedule_name = @ScheduleName,
    @enabled = 1,
    @freq_type = 8,          -- 8 = Semanalmente (Se usa para
definir días específicos)
    @freq_interval = 62,     -- Bitmask: 62 = 2+4+8+16+32 = Lun,
Mar, Mie, Jue, Vie, Sáb
    @freq_subday_type = 1,
    @freq_recurrence_factor = 1, -- Cada 1 semana
    @active_start_time = 233000, -- 11:30 PM
    @schedule_id = @schedule_id OUTPUT;

-- 5. Adjuntar la Programación
EXEC dbo.sp_attach_schedule @job_id = @jobId, @schedule_id =
@schedule_id;
```

```
-- 6. Habilitar el Job
EXEC dbo.sp_update_job @job_id = @jobId, @enabled = 1;
GO
```



3. Justificación Técnica de la Solución Aplicada

Concepto	Justificación Técnica
Backup Diferencial Diario	Ofrece un compromiso óptimo entre tiempo de backup (rápido, solo guarda cambios) y necesidad de espacio/archivos de restauración (solo se requiere el último FULL y el último DIFF).
Retención de 14 Días	Es un período de tiempo común que permite revertir cambios no deseados (ej. borrado accidental) hasta dos semanas atrás, manteniendo la base de datos `QhatuPeru` segura.
`sp_delete_backuphistory`	Esta es la forma **correcta y oficial** de limpiar el historial dentro de `msdb`. Mantener esta tabla limpia es vital para el rendimiento de SQL Server Agent y Management Studio.
`xp_cmdshell` y `DEL`	Se utiliza para eliminar los archivos físicos del sistema operativo, completando el ciclo de limpieza. Nota: Requiere configuración de seguridad avanzada (habilitar `xp_cmdshell`) y permisos de SO.
Programación 62	El valor bitmask `62` (Sábado + Viernes + Jueves + Miércoles + Martes + Lunes) asegura que el job **no se**

Concepto	Justificación Técnica
	ejecute el Domingo (valor 1), ya que el Domingo está reservado para el backup FULL.

4. Explicación de las Buenas Prácticas Utilizadas

- **Separación de Preocupaciones:** El Job se divide en dos pasos: **1. Respaldo** y **2. Limpieza**. Esto asegura que el backup se complete antes de intentar eliminar los archivos antiguos, lo cual es fundamental.
- **Limpieza Dual:** Se realiza la limpieza en **dos niveles**: el **historial de SQL Server** (`sp_delete_backuphistory`) y el **disco físico** (`xp_cmdshell`). Ambas son necesarias; una sin la otra deja registros en la BD o archivos huérfanos en el disco.
- **Uso de Compresión:** Se utiliza `WITH COMPRESSION` en el backup diferencial para ahorrar espacio en disco, lo cual es crítico en una estrategia de retención de 14 días.
- **Uso del Bitmask:** Se utiliza el Bitmask (`@freq_interval = 62`) en la programación para garantizar la precisión de la ejecución, excluyendo el día reservado para el backup FULL.

Códigos de Consulta y Verificación

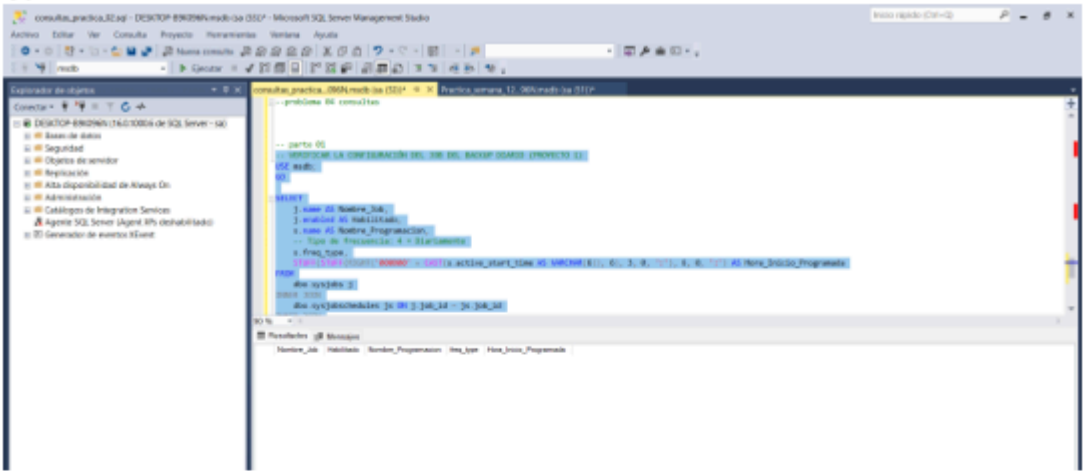
1. Proyecto 1: Backup FULL Diario

Este código se usa para verificar que el Job del backup diario se creó correctamente.

SQL

```
-- VERIFICAR LA CONFIGURACIÓN DEL JOB DEL BACKUP DIARIO (PROYECTO
1)
USE msdb;
GO

SELECT
    j.name AS Nombre_Job,
    j.enabled AS Habilitado,
    s.name AS Nombre_Programacion,
    -- Tipo de frecuencia: 4 = Diariamente
    s.freq_type,
    STUFF(STUFF(RIGHT('000000' + CAST(s.active_start_time AS
VARCHAR(6)), 6), 3, 0, ':'), 6, 0, ':') AS Hora_Inicio_Programada
FROM
    dbo.sysjobs j
INNER JOIN
    dbo.sysjobschedules js ON j.job_id = js.job_id
INNER JOIN
    dbo.sysschedules s ON js.schedule_id = s.schedule_id
```

```
WHERE
    j.name = 'Backup_QhatuPeru_Diario_FULL'; -- Asumiendo este
nombre
GO
```



2. Proyecto 2: Backup FULL Semanal + Auditoría

Estos códigos son para probar la ejecución manual del SP y verificar los registros en la tabla de auditoría.

SQL

```
-- 2.1. EJECUCIÓN MANUAL DEL BACKUP SEMANAL (PARA PRUEBA)
USE master;
GO

-- Esto ejecuta el backup FULL, la verificación y el registro en la
tabla de Auditoría.
EXEC dbo.sp_Backup_QhatuPeru_Semanal;
GO

-- 2.2. CONSULTA PARA VERIFICAR EL REGISTRO DE AUDITORÍA (PROYECTO
2)
-- (Muestra los 10 resultados más recientes del proceso)
USE QhatuPeru;
GO

SELECT TOP 10
    ID,
    Fecha_Ejecucion,
    Tipo_Backup,
    Estado_Backup,          -- Debería ser 'EXITOSO' o 'FALLIDO'
    Estado_Verificacion,    -- Debería ser 'OK' o 'ERROR'
    Mensaje_Verificacion    -- Mensaje de éxito o detalle del error
FROM
    dbo.Log_Backup_Auditoria
ORDER BY
    Fecha_Ejecucion DESC;
GO
```

3. Proyecto 3: Estrategia Combinada (FULL + DIFF + LOG)

Este código se usa para verificar el modo de recuperación, un requisito clave para el backup de LOGs.

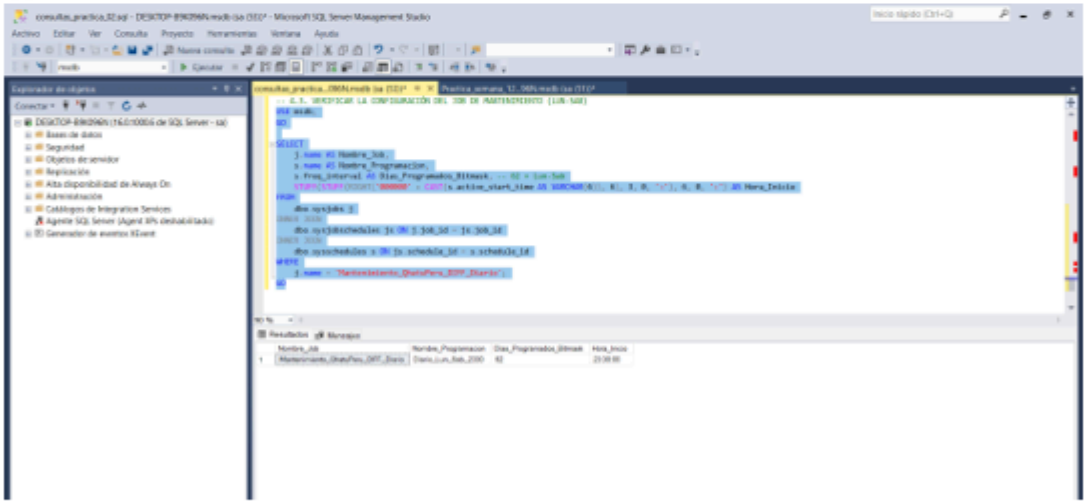
SQL
```
-- 3.1. VERIFICAR EL MODELO DE RECUPERACIÓN (DEBE SER 'FULL')
USE master;
GO

SELECT
    name AS Nombre_BD,
    recovery_model_desc AS Modelo_Recuperacion
FROM
    sys.databases
WHERE
    name = 'QhatuPeru';

-- ¡Debe decir 'FULL' para que el backup de LOG funcione!
GO

-- 3.2. EJECUCIÓN MANUAL DE LOS 3 TIPOS DE BACKUP (Para generar
archivos de prueba)
USE master;
GO
EXEC dbo.sp_Backup_QhatuPeru_FULL;
EXEC dbo.sp_Backup_QhatuPeru_DIFF;
EXEC dbo.sp_Backup_QhatuPeru_LOG;
GO
```

4. Proyecto 4: Estrategia Diferencial + Limpieza

Estas consultas son esenciales para verificar que los backups diferenciales se registran y que el proceso de limpieza y retención funciona.

SQL
```
-- 4.1. EJECUCIÓN MANUAL DEL PROCESO DE LIMPIEZA
-- Esto simulará el mantenimiento, eliminando el historial y los
archivos más antiguos de 14 días.
USE msdb;
GO
EXEC dbo.sp_Limpieza_Backups_Antiguos;
GO

-- 4.2. VERIFICAR EL HISTORIAL DE BACKUPS DIFERENCIALES Y
ANTIGÜEDAD
USE msdb;
GO

SELECT
    b.backup_start_date,
    b.physical_device_name,
    -- Tipo: I=Diferencial, D=FULL, L=LOG
    b.type AS Tipo_Backup,
```

```
    DATEDIFF(day, b.backup_finish_date, GETDATE()) AS
Dias_Antiguedad
FROM
    dbo.backupset b
WHERE
    b.database_name = 'QhatuPeru'
    AND b.type = 'I'
ORDER BY
    b.backup_finish_date DESC;
GO

-- 4.3. VERIFICAR LA CONFIGURACIÓN DEL JOB DE MANTENIMIENTO (LUN-
SAB)
USE msdb;
GO

SELECT
    j.name AS Nombre_Job,
    s.name AS Nombre_Programacion,
    s.freq_interval AS Dias_Programados_Bitmask, -- 62 = Lun-Sab
    STUFF(STUFF(RIGHT('000000' + CAST(s.active_start_time AS
VARCHAR(6)), 6), 3, 0, ':'), 6, 0, ':') AS Hora_Inicio
FROM
    dbo.sysjobs j
INNER JOIN
    dbo.sysjobschedules js ON j.job_id = js.job_id
INNER JOIN
    dbo.sysschedules s ON js.schedule_id = s.schedule_id
WHERE
    j.name = 'Mantenimiento_QhatuPeru_DIFF_Diario';
GO
```



Proyecto 5: Restauración en Distintos Escenarios

1. Enunciado del Ejercicio

Practicar restauraciones en 3 escenarios:

a) Restauración completa (recuperación estándar).

b) Restauración punto-en-el-tiempo por fallo humano (ejemplo: eliminación masiva accidental).

c) Restauración desde backup corrupto mediante verificación y usar backup alternativo.

2. Script de la Solución en T-SQL y Simulaciones

Para estos escenarios, se asume que la base de datos `QhatuPeru` está utilizando el **Modelo de Recuperación FULL** (Proyecto 3) y que se tienen archivos de backup FULL y de LOG recientes.

Escenario Setup: Generar Cambios y Archivos LOG

Para demostrar la restauración, necesitamos simular cambios y generar backups de LOG que los contengan.

SQL

```
-- *** PASO 0: GENERAR BACKUP BASE Y CAMBIOS ***
USE master;
GO

-- 0.1. Generar un Backup FULL base reciente
EXEC dbo.sp_Backup_QhatuPeru_FULL;
GO
-- ARCHIVO A: FullBackup_BASE.bak

-- 0.2. Generar algunos cambios de datos (simulación de actividad)
USE QhatuPeru;
INSERT INTO dbo.ARTICULO (Codigo, Descripcion, Precio) VALUES ('P5-
01', 'Articulo de Prueba', 10.00);
GO
-- ARCHIVO B: LogBackup_Cambio1.trn

-- 0.3. Generar el primer LOG
USE master;
EXEC dbo.sp_Backup_QhatuPeru_LOG;
GO

-- 0.4. Generar un cambio crítico (ej. el fallo humano,
eliminación)
USE QhatuPeru;
-- Suponemos que este es el punto que queremos evitar al restaurar
DELETE FROM dbo.ARTICULO WHERE Precio > 50;
GO
DECLARE @PuntoDeFallo DATETIME = GETDATE(); -- Capturar el tiempo
del fallo

-- 0.5. Generar el último LOG que contiene el fallo (pero aún no se
ha ejecutado)
USE master;
EXEC dbo.sp_Backup_QhatuPeru_LOG;
```

```
GO
-- ARCHIVO C: LogBackup_Fallo.trn
```

a) Restauración Completa (Recuperación Estándar)

Esto restaura la base de datos al estado exacto del último backup de LOG disponible.

SQL

```
-- *** ESCENARIO A: RESTAURACIÓN COMPLETA (Estado más reciente) ***
USE master;
GO
-- 1. Poner la BD en modo OFFLINE para restauración (si aún existe)
ALTER DATABASE QhatuPeru SET OFFLINE WITH ROLLBACK IMMEDIATE;
GO

-- 2. Restaurar el FULL base con NORECOVERY
RESTORE DATABASE QhatuPeru
FROM DISK =
'C:\SQL_Backups\FULL\QhatuPeru_FULL_AAAAMMDD_HHMMSS.bak' -- <--
Archivo A
WITH NORECOVERY, REPLACE;
GO

-- 3. Aplicar todos los LOGs subsiguientes (hasta el último
disponible)
RESTORE LOG QhatuPeru
FROM DISK =
'C:\SQL_Backups\LOG\QhatuPeru_LOG_AAAAMMDD_HHMMSS_Cambio1.trn' --
<-- Archivo B
WITH NORECOVERY;
GO

RESTORE LOG QhatuPeru
FROM DISK =
'C:\SQL_Backups\LOG\QhatuPeru_LOG_AAAAMMDD_HHMMSS_Fallo.trn' -- <--
Archivo C (Contiene el fallo)
WITH RECOVERY; -- RECOVERY: La base de datos queda ONLINE y lista
para usar.
GO
```

b) Restauración Punto-en-el-Tiempo (Fallo Humano)

Esto restaura la base de datos al estado **justo antes** de la hora de la eliminación masiva (el `@PuntoDeFallo` capturado arriba).

SQL

```
-- *** ESCENARIO B: RESTAURACIÓN PUNTO-EN-EL-TIEMPO (Pre-Fallo
Humano) ***
USE master;
GO
-- 1. Poner la BD en modo OFFLINE para restauración
ALTER DATABASE QhatuPeru SET OFFLINE WITH ROLLBACK IMMEDIATE;
GO

-- 2. Restaurar el FULL base con NORECOVERY
RESTORE DATABASE QhatuPeru
```

```
FROM DISK =
'C:\SQL_Backups\FULL\QhatuPeru_FULL_AAAAMMDD_HHMMSS.bak' -- <--
Archivo A
WITH NORECOVERY, REPLACE;
GO

-- 3. Aplicar los LOGs usando la cláusula STOPAT
-- La variable @PuntoDeFallo debe contener el valor capturado antes
del DELETE.
DECLARE @PuntoDeFallo VARCHAR(30) = '2025-11-25 21:05:00.000'; --
Ejemplo: La hora antes del delete

-- Aplicar el LOG que contiene el FALLO, pero detenerse justo antes
de la hora del fallo.
RESTORE LOG QhatuPeru
FROM DISK =
'C:\SQL_Backups\LOG\QhatuPeru_LOG_AAAAMMDD_HHMMSS_Fallo.trn' -- <--
Archivo C
WITH STOPAT = @PuntoDeFallo, RECOVERY; -- STOPAT: Detiene la
restauración antes de la hora especificada.
GO
-- Resultado: La base de datos queda ONLINE, y la eliminación
masiva NO se ejecuta.
```

c) Restauración desde Backup Corrupto (Verificación y Alternativa)

Este escenario demuestra la importancia del `RESTORE VERIFYONLY` (del Proyecto 1 y 2) para detectar un archivo dañado y la lógica para cambiar al último backup conocido como bueno.

SQL

```
-- *** ESCENARIO C: RESTAURACIÓN CON BACKUP CORRUPTO ***
USE master;
GO

-- 1. SIMULAR VERIFICACIÓN FALLIDA
-- (Supongamos que este es un backup LOG reciente, Archivo D)
RESTORE VERIFYONLY
FROM DISK = 'C:\SQL_Backups\LOG\QhatuPeru_LOG_Corrupto.trn';
GO
-- Resultado Esperado: Mensaje 3241 (The media family on device...
is incorrectly formed.)
-- Conclusión: Este backup no sirve.

-- 2. IDENTIFICAR ÚLTIMO BACKUP CONOCIDO COMO BUENO
-- Se usa la tabla msdb.dbo.backupset para encontrar el anterior al
corrupto.

-- 3. EJECUTAR RESTAURACIÓN UTILIZANDO EL ARCHIVO ALTERNATIVO BUENO
-- Si el último LOG (Archivo D) está corrupto, debemos restaurar
hasta el LOG (Archivo C) anterior a ese.
ALTER DATABASE QhatuPeru SET OFFLINE WITH ROLLBACK IMMEDIATE;
GO

-- Restaurar la cadena (FULL -> DIFF -> LOG anterior)
-- ... [Comandos FULL y DIFF como en Escenario A/B] ...

-- Aplicar el último LOG CONOCIDO COMO BUENO (Archivo C)
```

```
RESTORE LOG QhatuPeru
FROM DISK =
'C:\SQL_Backups\LOG\QhatuPeru_LOG_AAAAMMDD_HHMMSS_Fallo.trn' -- <--
El último bueno
WITH RECOVERY; -- Se usa RECOVERY aquí para aceptar una pequeña
pérdida de datos, pero mantener la integridad.
GO
-- Resultado: La BD está en línea, se pierde la data entre el LOG
bueno y el LOG corrupto, pero se mantiene la integridad de la base.
```

3. Justificación Técnica de la Solución Aplicada

Escenario	Técnica Clave	Justificación
a) Completa	`RESTORE ... WITH RECOVERY`	Finaliza la cadena de restauración, aplica todos los logs disponibles, y pone la BD en estado operacional, logrando el **RPO** (Punto de Recuperación) más bajo posible.
b) Punto-en-el-Tiempo	`RESTORE LOG ... WITH STOPAT = 'datetime'`	Utiliza el LOG de transacciones para detener la aplicación de cambios en una marca de tiempo precisa. Es esencial para recuperaciones por **fallo humano** o lógico (ej. `UPDATE` o `DELETE` masivo accidental).
c) Backup Corrupto	`RESTORE VERIFYONLY` & **Retención**	`VERIFYONLY` (Proyecto 1) confirma la validez de un archivo antes de usarlo. Si falla, el DBA debe tener una **retención** suficiente (Proyecto 4) para cambiar a la copia de backup inmediatamente anterior conocida como buena, minimizando la pérdida.

4. Explicación de las Buenas Prácticas Utilizadas

Minimizar Pérdida de Datos (RPO): Los backups de LOGs continuos (Proyecto 3) son la base de los escenarios B y C, ya que permiten la recuperación granular al punto exacto del fallo o al último punto conocido como bueno.
Aislamiento de Restauración: El uso de `ALTER DATABASE ... SET OFFLINE WITH ROLLBACK IMMEDIATE` antes de restaurar es una práctica estándar para asegurar que no haya conexiones activas que interfieran con el proceso de restauración.
Uso de `REPLACE` (Opcional): Se usa `WITH REPLACE` en el primer comando `FULL` de restauración para forzar la sobrescritura de la base de datos existente, lo cual es necesario en entornos de prueba o si la base de datos está corrupta.
Cambio de Estrategia: En el Escenario C, la buena práctica es **aceptar la pérdida mínima de datos** (los cambios entre el último backup bueno y el fallo) a cambio de la **integridad y disponibilidad** de la base de datos.