Tema 3: Backup y Restore. Backup en Línea.

1. Introducción

En esta parte del informe abordaremos el proceso completo de backup y restore en SQL Server, utilizando el modelo de recuperación FULL, el cual permite realizar backups en línea, es decir, mientras la base se encuentra operativa.

El trabajo consiste en generar una secuencia de respaldos (FULL → LOG1 → LOG2), realizar inserciones en una tabla de referencia y recuperar la base en distintos puntos del tiempo para verificar la correcta integridad de los datos.
Este enfoque garantiza la continuidad operativa y demuestra cómo SQL Server permite restaurar estados anteriores sin pérdida de información, siempre que se respete el orden de la cadena de backup.

2. Objetivos

- Conocer los distintos tipos de backup y su uso dentro del modelo de recuperación FULL.
- Implementar una estrategia de respaldo basada en una cadena de restauración (FULL + LOGs).
- Verificar la consistencia de los datos luego de cada restauración.
- Comprender el funcionamiento del backup en línea y su importancia operativa.

3. Conceptos Fundamentales

Modelo de Recuperación FULL

SQL Server permite definir el *modelo de recuperación* de una base para determinar cómo se gestionan los cambios y qué tipo de recuperaciones son posibles.
Los modelos más comunes son: SIMPLE, FULL y BULK_LOGGED.

El modelo FULL registra todas las operaciones en el log de transacciones, lo que permite:

- Hacer backup del log.
- Recuperar la base hasta un punto específico en el tiempo.
- Restaurar estados intermedios mediante secuencias FULL → LOG1 → LOG2 → LOGn.
- Realizar backups en línea sin interrumpir la operación de usuarios.

Este modelo se habilita con:

ALTER DATABASE sistema_riego

SET RECOVERY FULL;

En este trabajo se utiliza FULL porque es el único que permite restauraciones con múltiples logs, recreando estados intermedios sin pérdida de datos.

El Log de Transacciones: cómo funciona

El Transaction Log o archivo de log funciona como un registro continuo de:

- inserciones,
- actualizaciones,
- eliminaciones,
- creación de objetos,
- transacciones confirmadas y no confirmadas.

Cada operación genera un número interno llamado LSN (Log Sequence Number).
Este valor ordena cronológicamente todos los cambios de la base.

El proceso de restauración exige que los backups se apliquen respetando el orden de LSN. De lo contrario puede surgir un error como nos ocurrió durante el intento de este proyecto que surgió el mensaje:

> "The log in this backup set begins at LSN… which is too recent…"

que indica que se está intentando restaurar un log que no continúa la secuencia.

Backup en Línea

El backup en línea es un respaldo que se ejecuta mientras la base está siendo utilizada por usuarios o aplicaciones, sin necesidad de detenerla ni ponerla en modo exclusivo.

✔ La base sigue operativa
✔ Los cambios continúan registrándose
✔ El log de transacciones asegura consistencia

SQL Server garantiza que el backup sea coherente porque:

1. Congela páginas de datos momentáneamente para capturarlas.
2. Utiliza el log para completar las transacciones en curso.
3. Crea un snapshot interno del estado del archivo en el instante del backup.

El backup en línea solo es posible si el modelo de recuperación es FULL o BULK_LOGGED.

Comandos utilizados en un backup en línea

Backup Full (en línea):

BACKUP DATABASE sistema_riego

TO DISK = 'C:\Backups\sistema_riego_FULL.bak'

WITH INIT, NAME = 'Backup FULL';

- Realiza copia completa de los archivos .mdf y .ldf.
- No interrumpe el uso de la base.

Backup del Log (en línea)

BACKUP LOG sistema_riego

TO DISK = 'C:\Backups\sistema_riego_LOG1.trn'

WITH INIT, NAME = 'Backup LOG 1';

El backup de log registra únicamente:

- Cambios desde el anterior backup de log o full.
- Los LSN que continuarán la secuencia de restauración.

Restauración con NORECOVERY y RECOVERY

Cuando se restaura una base, SQL Server debe saber si:

- va a seguir recibiendo logs, o
- ya se terminó la secuencia.

NORECOVERY: Mantiene la base en estado RESTORING, permitiendo aplicar backups de log posteriores.

RESTORE DATABASE sistema_riego

FROM DISK='FULL.bak'

WITH NORECOVERY;

Mientras está en RESTORING:

- No se puede consultar la base
- No se puede modificar
- No se puede cambiar a MULTI_USER
- Solo se puede seguir aplicando backups

RECOVERY: Cierra la secuencia de restauración y deja la base ONLINE.

RESTORE LOG sistema_riego

FROM DISK='LOG1.trn'

WITH RECOVERY;

Una vez que se usa RECOVERY:

- La restauración se da por finalizada
- No se pueden aplicar más logs (se rompería la cadena)

Secuencia Completa del proceso de Backup / Restore

Backup

1. FULL
2. Generación de nuevos datos
3. LOG1
4. Generación de más datos
5. LOG2

(Lo desarrollamos en el punto 4).

Restore

- Estado 1: Solo FULL
- Estado 2: FULL + LOG1
- Estado 3: FULL + LOG1 + LOG2

(Lo desarrollamos en el punto 5).

Errores típicos en la cadena de restauración

1) Usar RECOVERY en el momento incorrecto

Cierra la secuencia e impide aplicar logs posteriores.

2) Restaurar un FULL que no corresponde al LOG

Genera error de LSN no coincidente.

3) Intentar restaurar con la base ONLINE

Da error: "database is in use".

4) Querer cambiar MULTI_USER mientras está en RESTORING

Da error: “ALTER DATABASE is not permitted while… restoring”

¿Por qué es útil el backup en línea?

En entornos reales:

- No se puede detener la base para respaldarla.
- Los usuarios deben operar mientras se realiza el backup.
- Debe garantizarse la consistencia aunque existan transacciones abiertas.

El proceso FULL + LOG permite:

- Mantener un historial detallado de cambios.
- Restaurar estados intermedios según necesidad.
- Proteger la integridad de los datos ante fallas.

4. Desarrollo

A continuación se documenta todo el proceso solicitado en la consigna.

Creación de tablas y carga de datos de prueba

Para realizar las pruebas se creó la tabla ZonaRiego y un procedimiento almacenado GenerarZonasRiegoPrueba, que permite generar N registros automáticos:

```
CREATE OR ALTER PROCEDURE GenerarZonasRiegoPrueba
    @Cantidad INT = 100
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @i INT = 1;

    WHILE @i <= @Cantidad
    BEGIN
        INSERT INTO ZonaRiego (nombre, descripcion, superficie)
        VALUES (
            CONCAT('Zona ', RIGHT('000' + CAST(@i AS varchar(3)), 3)),
            CONCAT('Zona de riego de prueba ', @i),
            CAST(10 + (ABS(CHECKSUM(NEWID())) % 991) AS float)
        );

        SET @i = @i + 1;
    END;
END;
```

Mediante la iteración desde i = 1 hasta la cantidad necesaria inserta los registros con nombre y descripción iterativos y superficies generadas de manera aleatoria.

Con el siguiente código se genero 100 registros para la base inicial:

EXEC GenerarZonasRiegoPrueba @Cantidad = 100;

```
--Todos los insertados
SELECT *
FROM ZonaRiego;
GO
```

Results | Messages

	id_zona	nombre	descripcion	superficie
1	1	Zona 001	Zona de riego de prueba 1	139
2	2	Zona 002	Zona de riego de prueba 2	951
3	3	Zona 003	Zona de riego de prueba 3	34
4	4	Zona 004	Zona de riego de prueba 4	38
5	5	Zona 005	Zona de riego de prueba 5	863
6	6	Zona 006	Zona de riego de prueba 6	130
7	7	Zona 007	Zona de riego de prueba 7	835
8	8	Zona 008	Zona de riego de prueba 8	706

R9PAEE (16.0 RTM) | DESKTOP-8R9PAEE\Kevin ... | sistema_riego | 00:00:00 | 100 rows

Se verificó los 100 registros ingresados con la zona, descripción y superficie.

Configuración del modelo de recuperación

Primero verificamos que el modelo de recuperación este en FULL.

```
-- Vemos el modelo de recuperación actual --
SELECT name, recovery_model_desc
FROM sys.databases
WHERE name = 'sistema_riego';
GO
```

Results | Messages

	name	recovery_model_desc
1	sistema_riego	FULL

Sino lo cambiamos con:

ALTER DATABASE sistema_riego

SET RECOVERY FULL;

Realización del Backup FULL

Realizamos el backup full mediante el siguiente código:

```
BACKUP DATABASE sistema_riego
TO DISK = 'C:\Backups\sistema_riego_FULL.bak'
WITH INIT,                    -- sobrescribe el archivo si existe
     NAME = 'Backup FULL inicial sistema_riego';
GO
```

```
Messages
Processed 512 pages for database 'sistema_riego', file 'sistema_riego' on fil
Processed 1 pages for database 'sistema_riego', file 'sistema_riego_log' on f
BACKUP DATABASE successfully processed 513 pages in 0.041 seconds (97.596 MB/
```

Q... | DESKTOP-8R9PAEE (16.0 RTM) | DESKTOP-8R9PAEE\Kevin ... | sistema_riego | 00:00:00 | 0 rows

Y vemos que se realiza de forma exitosa.

Primeros 10 inserts y verificación

Mediante el procedimiento GenerarZonasRiegoPrueba insertamos 10 registros:

```
EXEC GenerarZonasRiegoPrueba @Cantidad = 10;
GO
```

Y verificamos:

```
SELECT COUNT(*) AS CantZonaRiego
FROM ZonaRiego;
GO
```

Results | Messages

	CantZonaRiego
1	110

Q... | DESKTOP-8R9PAEE (16.0 RTM) | DESKTOP-8R9PAEE\Kevin ... | sistema_riego | 00:00:00 | 1 rows

Backup del archivo de LOG (LOG1)

Realizamos el backup LOG1 junto con la hora del backup para su registro:

```
BACKUP LOG sistema_riego
TO DISK = 'C:\Backups\sistema_riego_LOG1.trn'
WITH INIT,
     NAME = 'Backup LOG 1 - luego de primeros 10 inserts en ZonaRiego';
GO

SELECT SYSDATETIME() AS HoraBackupLog1; --Guardamos la hora del backup
GO
```

Results | Messages

	HoraBackupLog1
1	2025-11-17 11:54:27.7309108

Segundos 10 inserts y verificación

Mediante el procedimiento GenerarZonasRiegoPrueba insertamos otros 10 registros:

```
EXEC GenerarZonasRiegoPrueba @Cantidad = 10;
GO
```

Y verificamos:

```
SELECT COUNT(*) AS CantZonaRiego
FROM ZonaRiego;
GO
```

Results | Messages

	CantZonaRiego
1	120

Segundo Backup de Log (LOG2)

Realizamos el backup LOG2 junto con la hora del backup para su registro:

```
BACKUP LOG sistema_riego
TO DISK = 'C:\Backups\sistema_riego_LOG2.trn'
WITH INIT,
     NAME = 'Backup LOG 2 - luego de otros 10 inserts en ZonaRiego';
GO

SELECT SYSDATETIME() AS HoraBackupLog1; --Guardamos la hora del backup
GO
```

Results | Messages

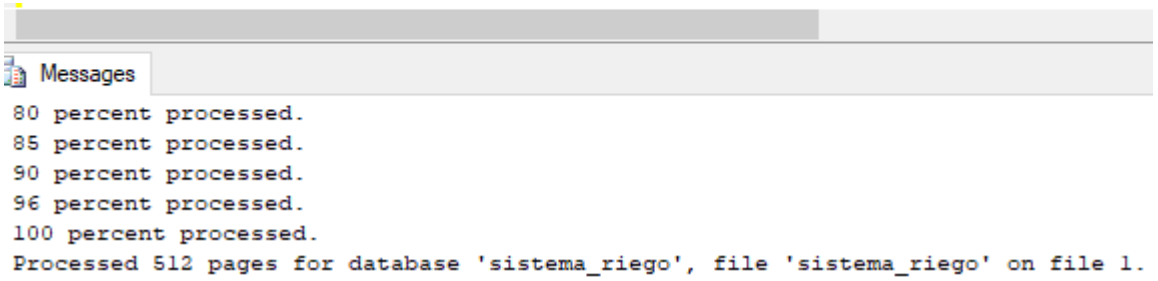
HoraBackupLog1
2025-11-17 11:57:47.6365938

5. Restauraciones Solicitadas

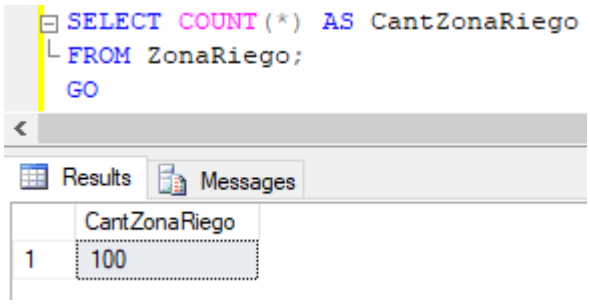
Restauración del FULL

Cambiamos a Master (USE Master) y ejecutamos:

```
ALTER DATABASE sistema_riego
SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
GO

RESTORE DATABASE sistema_riego
FROM DISK = 'C:\Backups\sistema_riego_FULL.bak'
WITH RECOVERY,
     REPLACE,
     STATS = 5;
GO

ALTER DATABASE sistema_riego
SET MULTI_USER;
GO
```



Resultado esperado: 100 registros. Cambiamos a sistema_riego (USE) y verificamos:



Obteniendo un resultado exitoso.

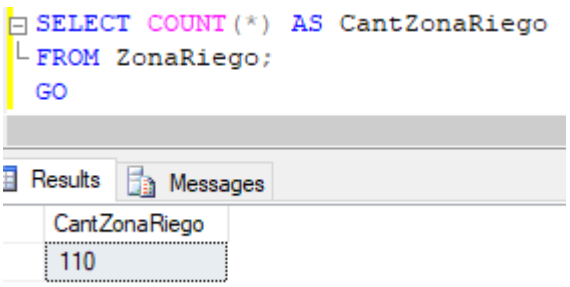
Restauración FULL + LOG1

```
USE master;
```

```
go

ALTER DATABASE sistema_riego
SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
GO

-- Restauramos FULL (sin recuperar todavia) --
RESTORE DATABASE sistema_riego
FROM DISK = 'C:\Backups\sistema_riego_FULL.bak'
WITH NORECOVERY,
     REPLACE,
     STATS = 5;
GO

-- Aplicamos LOG1 y RECOVERY --
RESTORE LOG sistema_riego
FROM DISK = 'C:\Backups\sistema_riego_LOG1.trn'
WITH RECOVERY,                -- La base queda ONLINE
     STATS = 5;
GO

ALTER DATABASE sistema_riego
SET MULTI_USER;
GO

USE sistema_riego;
GO
```

Realizamos el backup de manera secuencial, empezando por el base (FULL) y luego el LOG1. Para ello, como vimos antes, respetamos la cadena ejecutando el full con NORECOVERY y RECOVERY en LOG1.

Resultado esperado: 110 registros. Verificamos:



Resultado exitoso.

Restauración FULL + LOG1 + LOG2

Siguiendo con la restauración anterior, para restaurar el LOG2 lo hacemos de manera secuencial

```
ALTER DATABASE sistema_riego
SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
GO

RESTORE DATABASE sistema_riego
FROM DISK = 'C:\Backups\sistema_riego_FULL.bak'
WITH NORECOVERY,
     REPLACE,
     STATS = 5;
GO

RESTORE LOG sistema_riego
FROM DISK = 'C:\Backups\sistema_riego_LOG1.trn'
WITH NORECOVERY,
     STATS = 5;
GO

RESTORE LOG sistema_riego
FROM DISK = 'C:\Backups\sistema_riego_LOG2.trn'
WITH RECOVERY,
     STATS = 5;
GO
```

Realizamos el backup, empezando por el base (FULL), luego LOG1 Y finalmente LOG2. Para ello, como vimos antes, respetamos la cadena ejecutando el full con NORECOVERY , NORECOVERY en LOG1 y RECOVERY en LOG2.

Resultado esperado: 120 registros. Verificamos:

```
SELECT COUNT(*) AS CantZonaRiego
FROM ZonaRiego;
GO
```

Results | Messages

	CantZonaRiego
1	120

Resultado exitoso.

6. Conclusiones

A partir de las pruebas realizadas se concluye que:

1. El modelo de recuperación FULL es esencial para permitir backups en línea y restauraciones detalladas mediante archivos de log.
2. La estrategia de respaldo utilizada demostró ser consistente:
    - El estado inicial (100 registros) fue restaurado correctamente desde el FULL.
    - Las restauraciones intermedias utilizando backups de LOG permitieron recuperar estados exactos después de los primeros 10 y 20 inserts.
3. Se comprobó que SQL Server aplica estrictamente la cadena de restauración: si se intenta restaurar un log que no corresponde al LSN adecuado, el motor devuelve un error, protegiendo la integridad del sistema.
4. El proceso realizado garantiza que en un entorno real se podría recuperar la base de datos sin pérdida de información, incluso si ocurre un fallo entre inserciones.
5. La documentación generada, junto con las evidencias gráficas, demuestra la correcta implementación de la secuencia de backup y restore solicitada.