Tecnologías de la Información, Ingeniería en Desarrollo y Gestión de Software

Store para creación de backups

Asignatura: Administración de base de datos

17 DE MARZO DE 2025

BENJAMIN PEÑA MARIN
Grupo. 1
Docente: José Herrera Gallardo

SQL dinamico logins

1. Introducción

La protección y recuperación de la información son aspectos críticos en la administración de bases de datos. En este proyecto se desarrolló una solución integral para gestionar los backups en SQL Server, que garantiza la disponibilidad y resguardo de los datos mediante la automatización de procesos. La solución contempla:

- La creación de un stored procedure que genera backups de bases de datos, diferenciando entre backups completos, diferenciales y de logs.
- La integración de esta funcionalidad en una aplicación web desarrollada en ASP.NET Core (.NET 8), que permite ejecutar backups manualmente y visualizar los archivos generados.
- La parametrización de la ubicación de almacenamiento de backups para funcionar en entornos físicos y en contenedores Docker mediante el uso de volúmenes.
- La automatización de la ejecución del stored procedure a través de SQL Server Agent, configurando tres jobs con frecuencias específicas: backup completo semanal, backup diferencial diario y backup de logs cada 15 minutos.

Este informe describe el proceso de diseño, desarrollo e integración de la estrategia de backups, así como las pruebas realizadas para validar su correcto funcionamiento en ambos entornos.

2. Objetivos

Objetivo General:
Automatizar la generación y gestión de backups en SQL Server, integrando la solución en una aplicación ASP.NET Core que opere tanto en entornos locales como en contenedores Docker, y programar su ejecución automática mediante SQL Server Agent.

Objetivos Específicos:

- Desarrollar un stored procedure en SQL Server que, de manera dinámica y segura, genere backups según el tipo (FULL, DIFFERENTIAL o LOG) de una base de datos.
- Implementar un servicio en la aplicación ASP.NET Core que invoque dicho stored procedure y que, mediante la lectura de la configuración, determine la ubicación de almacenamiento de backups:
    - En la máquina física, utilizando una ruta local (por ejemplo, C:\Backups).
    - En un contenedor Docker, utilizando una ruta correspondiente al volumen montado (por ejemplo, /backups).
- Configurar SQL Server Agent para programar la ejecución automática del stored procedure, estableciendo:
    - Backup completo semanal.
    - Backup diferencial diario.
    - Backup de logs cada 15 minutos.
- Realizar pruebas en ambos entornos para validar la correcta creación, almacenamiento y programación de backups.

3. Desarrollo

3.1. Diseño del Stored Procedure para Backups

El stored procedure sp_BackupDatabase fue desarrollado con los siguientes aspectos clave:

- **Recepción de Parámetros:**
  Recibe como parámetros el nombre de la base de datos (@DatabaseName) y el tipo de backup (@BackupType), permitiendo generar el backup según el valor proporcionado (FULL, DIFFERENTIAL o LOG).
- **Definición de Rutas y Creación de Carpetas:**
  Se establece una ruta raíz de backups (definida en la lógica de la aplicación) y, en función del tipo de backup, se determina una subcarpeta (por ejemplo, Full, Differential o Log).
  Se utiliza xp_cmdshell para crear la carpeta si no existe, garantizando que el backup se almacene en la ubicación correcta.
- **Generación de Nombre de Archivo:**
  Se construye un nombre de archivo único incorporando el nombre de la base de datos, el tipo de backup y un timestamp. Esto evita sobrescrituras y facilita la identificación de cada respaldo.
- **Ejecución y Manejo de Errores:**
  Se utiliza un bloque TRY...CATCH para ejecutar el comando de backup (usando las instrucciones BACKUP DATABASE o BACKUP LOG) y capturar errores, devolviendo mensajes descriptivos en caso de incidencias.

3.2. Integración con la Aplicación y el Entorno Docker

La aplicación ASP.NET Core se diseñó para integrarse con la solución de backups mediante un servicio denominado BackupService, que cumple las siguientes funciones:

- **Lectura de Configuración:**
  El servicio lee la sección BackupSettings del archivo *appsettings.json* para determinar la ruta de almacenamiento. Según el parámetro "CurrentEnvironment", se utiliza:
  - La ruta local (LocalPath) si se ejecuta en la máquina física.
  - La ruta definida para Docker (DockerPath) si la aplicación se despliega en un contenedor con el volumen montado.
- **Ejecución del Stored Procedure:**
  El método ExecuteBackupAsync se encarga de abrir una conexión a SQL Server, invocar el stored procedure sp_BackupDatabase y pasar los parámetros necesarios para generar el backup.
- **Listado de Backups:**
  El método GetBackupsList recorre la estructura de directorios de la ruta de backups y genera una lista de modelos (BackupModel), que luego se muestra

en la interfaz de la aplicación para permitir la visualización y descarga de archivos.

Esta parametrización permite que, al ejecutar la aplicación en Docker, los backups se almacenen en la ruta del contenedor (/backups), garantizando persistencia mediante un volumen montado.

3.3. Programación de Jobs en SQL Server Agent

Para automatizar la creación de backups, se configuraron tres jobs en SQL Server Agent:

- **Job de Backup Completo:**
  - **Frecuencia:** Semanal.
  - **Acción:** Ejecuta el stored procedure con @BackupType = 'FULL' para generar un backup completo de la base de datos.
- **Job de Backup Diferencial:**
  - **Frecuencia:** Diario.
  - **Acción:** Ejecuta el stored procedure con @BackupType = 'DIFFERENTIAL' para generar un backup diferencial.
- **Job de Backup de Logs:**
  - **Frecuencia:** Cada 15 minutos.
  - **Acción:** Ejecuta el stored procedure con @BackupType = 'LOG' para generar un backup de logs de transacciones.

Cada job se configuró mediante scripts T-SQL que asignan un schedule específico y vinculan la ejecución del stored procedure a la frecuencia requerida, garantizando la actualización constante y la protección de la información.

4. Dificultades Encontradas y Soluciones

1. **Parametrización de la Ruta de Backups en Diferentes Entornos:**
   - **Dificultad:**
     La aplicación debía gestionar de forma transparente la ubicación de almacenamiento de backups, dependiendo de si se ejecutaba en una máquina física o en un contenedor Docker.
   - **Solución:**
     Se implementó la lectura de la configuración en *appsettings.json* mediante la sección BackupSettings, asignando dinámicamente la ruta a utilizar en el servicio BackupService.
2. **Integración del Stored Procedure con la Aplicación:**
   - **Dificultad:**
     Garantizar que el stored procedure se ejecute correctamente desde la aplicación y que los archivos se creen en la ubicación configurada.
   - **Solución:**
     Se desarrolló un servicio robusto que utiliza ADO.NET para ejecutar el stored procedure y se realizaron pruebas en ambos entornos para validar la creación de carpetas y archivos.
3. **Programación Automática de Jobs en SQL Server Agent:**
   - **Dificultad:**
     Configurar la ejecución periódica del stored procedure para distintos tipos de backup y asegurar que los jobs se ejecuten sin errores.
   - **Solución:**
     Se crearon tres jobs con scripts T-SQL específicos, asignándoles schedules distintos para backups completos, diferenciales y de logs. Se revisaron los historiales de ejecución y se ajustaron los parámetros de reintento para garantizar la fiabilidad.
4. **Persistencia en Contenedores Docker:**
   - **Dificultad:**
     Al desplegar la aplicación en Docker, era necesario asegurar que los backups se guardaran en un volumen persistente y no se perdieran al reiniciar el contenedor.
   - **Solución:**
     Se creó un contenedor especializado para almacenamiento de backups, montando un volumen en la ruta /backups. La aplicación, configurada para usar dicha ruta en modo Docker, garantiza que los archivos se almacenen de forma persistente.

5. Conclusiones

- **Automatización Efectiva:**
  La implementación del stored procedure y la integración mediante el servicio en ASP.NET Core han permitido automatizar la generación de backups, reduciendo la intervención manual y minimizando errores.
- **Flexibilidad y Escalabilidad:**
  La solución es adaptable a distintos entornos gracias a la parametrización de la ruta de almacenamiento. Al emplear Docker, se garantiza la persistencia de los archivos de backup en contenedores, facilitando la gestión en infraestructuras modernas.
- **Seguridad y Robustez:**
  El uso de validaciones, la generación dinámica de nombres de archivo y el manejo adecuado de errores aseguran que los backups se realicen de forma segura y confiable, protegiendo la integridad de los datos.
- **Programación Automática con SQL Server Agent:**
  La creación de jobs para ejecutar backups completos, diferenciales y de logs en intervalos específicos permite mantener actualizada la estrategia de respaldo sin intervención manual, lo que resulta crucial en entornos empresariales.

En conjunto, la solución implementada mejora significativamente la gestión y protección de la información en SQL Server, ofreciendo una herramienta flexible y robusta que se integra tanto en entornos tradicionales como en infraestructuras basadas en Docker.