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

# Store para creación de BD dinámico

Asignatura: Administración de base de datos


10 DE MARZO DE 2025

BENJAMIN PEÑA MARIN
Grupo. 1
Docente: José Herrera Gallardo

# SQL dinamico

Introducción ..... 3
**Objetivos** ..... 4
**Desarrollo** ..... 5
1. Diseño y Desarrollo del Stored Procedure ..... 5
2. Implementación de la Aplicación PHP ..... 5
3. Integración y Validación ..... 6
**Dificultades Encontradas y Soluciones** ..... 7
1. Errores de Sintaxis en SQL Dinámico ..... 7
2. Manejo de Parámetros Opcionales ..... 7
3. Integración y Validación en la Aplicación PHP ..... 7
4. Verificación y Depuración ..... 8
**Conclusiones** ..... 8

## Introducción

Comenzare recalcando la importancia de la administración de bases de datos que se ha convertido en un componente muy importante para el correcto funcionamiento de sistemas empresariales y aplicaciones web. El presente proyecto surge con el objetivo de automatizar y optimizar la creación de bases de datos en SQL Server, permitiendo la configuración dinámica de sus parámetros, así como la creación de bases de datos de manera dinamica asociados. Para ello, se desarrolló un procedimiento almacenado que facilita la creación de una base de datos con su archivo principal (filegroup PRIMARY) y la opción de agregar de forma opcional un filegroup secundario, brindando así una mayor flexibilidad en la asignación de recursos de almacenamiento.

La solución propuesta se complementa con el desarrollo de una aplicación web en PHP, la cual integra de manera intuitiva y profesional (empleando Bootstrap para un diseño moderno) un formulario que interactúa directamente con el procedimiento almacenado. Esta aplicación permite la creación de bases de datos de forma automatizada en SQL Server.

Durante el desarrollo del proyecto se abordaron diversos desafíos, tales como la generación dinámica y segura de sentencias SQL, la validación de parámetros opcionales, y la correcta integración de los componentes a través de conexiones seguras. La utilización de funciones como QUOTENAME() y el manejo adecuado de errores mediante bloques TRY...CATCH en SQL Server fueron determinantes para garantizar la robustez del sistema. Asimismo, la aplicación PHP fue diseñada para capturar y manejar las respuestas del servidor, ofreciendo al usuario retroalimentación clara sobre la ejecución de cada operación.

Este informe detalla el proceso de diseño, desarrollo e integración de la solución, las dificultades encontradas y las soluciones aplicadas, demostrando que la automatización de procesos en la administración de bases de datos no solo es posible, sino que también aporta un valor significativo en términos de eficiencia y seguridad.

## Objetivos

- **Objetivo:**
  Automatizar y optimizar el proceso de creación de bases de datos en SQL Server mediante el desarrollo de un procedimiento almacenado que permita configurar de forma dinámica todos los parámetros necesarios, y facilitar su integración en una aplicación web PHP para una administración más eficiente y segura.
- **Objetivos Específicos:**
  - Desarrollar un procedimiento almacenado que permita la creación de una base de datos con su archivo principal (filegroup PRIMARY) y la opción de agregar un filegroup secundario de forma opcional.
  - Permitir la configuración dinámica de parámetros críticos, tales como la ubicación, tamaño inicial, crecimiento y tamaño máximo de los archivos de datos y log.
  - Garantizar la robustez y seguridad del procedimiento mediante el uso de funciones como QUOTENAME() y el manejo adecuado de errores con bloques TRY...CATCH.
  - Integrar el procedimiento almacenado en una aplicación web desarrollada en PHP, que ofrezca una interfaz intuitiva y profesional (usando Bootstrap) para facilitar la administración de bases de datos.
  - Minimizar la intervención manual y reducir la posibilidad de errores en la creación y configuración de bases de datos en entornos empresariales.

**Desarrollo**

1. Diseño y Desarrollo del Stored Procedure

Para automatizar y optimizar la creación de bases de datos en SQL Server, se desarrolló el procedimiento almacenado sp_CrearBaseDeDatos. Este procedimiento fue concebido con el fin de permitir la configuración dinámica de los principales parámetros de una base de datos, tales como:

- **Nombre de la Base de Datos:** Se recibe como parámetro para crear un objeto único.
- **Ubicación y Configuración de Archivos:** Se establecen parámetros para la ubicación física y el tamaño inicial de los archivos de datos (MDF) y de log (LDF), así como sus opciones de crecimiento y tamaño máximo.
- **Filegroup Secundario (Opcional):** Además del filegroup PRIMARY, se incorporó la posibilidad de agregar un filegroup secundario. Este bloque se concatena de manera condicional, solo si se especifican los parámetros correspondientes (nombre y ruta del archivo).

*Aspectos Técnicos Destacados*

- **SQL Dinámico y Uso de QUOTENAME():**
  Se utilizó SQL dinámico para construir la sentencia CREATE DATABASE de manera flexible, permitiendo que los parámetros se inserten dinámicamente en la consulta. La función QUOTENAME() fue esencial para proteger los nombres de objetos, evitando problemas de inyección SQL y errores de sintaxis al incluir caracteres especiales o espacios.
- **Manejo de Parámetros Opcionales:**
  Se implementaron condicionales para evaluar si se han proporcionado los datos para el filegroup secundario. Si estos parámetros son nulos, el procedimiento omite la creación de dicho filegroup, garantizando así que la sentencia final sea válida sin introducir bloques incompletos o errores de sintaxis.
- **Control de Errores:**
  El procedimiento incorpora bloques TRY...CATCH que permiten capturar y gestionar errores durante la ejecución. De este modo, si ocurre algún fallo (por ejemplo, si la base de datos ya existe), se imprime un mensaje descriptivo y se retorna un código de error. Esta estrategia asegura que el proceso sea robusto y que se notifique de manera clara cualquier incidencia.

2. Implementación de la Aplicación PHP

La integración del stored procedure en la aplicación PHP fue un paso clave para automatizar la administración de bases de datos. La aplicación fue desarrollada con un enfoque modular y utilizando **Bootstrap** para lograr un diseño profesional y responsivo.

*Componentes Principales*

- **Conexión a SQL Server:**
  Se creó un archivo config.php que establece la conexión a SQL Server utilizando la extensión sqlsrv de PHP. Este archivo centraliza la configuración de conexión (servidor, base de datos por defecto, conjunto de caracteres y opciones de seguridad).
- **Formulario de Entrada:**
  Se desarrolló un formulario web que permite al usuario ingresar todos los parámetros necesarios para la creación de una base de datos: nombre de la BD, rutas de archivos, tamaños, crecimientos, etc. Además, se incluyeron campos para el filegroup secundario de manera opcional. El formulario fue diseñado con las clases de Bootstrap, lo que garantiza una interfaz limpia, intuitiva y moderna.
- **Consumo del Stored Procedure:**
  Una vez enviado el formulario, la aplicación recoge los datos y los envía de forma segura al servidor mediante una consulta parametrizada. La llamada se realiza utilizando la sintaxis "{CALL sp_CrearBaseDeDatos(?,?,?,?,?,?,?,?,?,?,?,?,?)}" y pasando un arreglo de parámetros que coincide con el orden y tipo de los definidos en el SP. Esto asegura que la inyección de datos se realice de forma segura.
- **Manejo de Respuestas y Mensajes:**
  La aplicación procesa la respuesta de la ejecución del SP. Si la ejecución es exitosa, se muestra un mensaje de éxito; de lo contrario, se captura el error y se muestra un mensaje descriptivo. Para facilitar la depuración, se incluyeron declaraciones PRINT @SQL; en el SP que permiten conocer la sentencia final generada (aunque estos mensajes se pueden ocultar en producción).

3. Integración y Validación

Una vez desarrollados tanto el stored procedure como la aplicación PHP, se realizaron pruebas de integración para validar que:

- **La Base de Datos se Creara Correctamente:**
  Se verificó que la ejecución del SP creara la base de datos en SQL Server con el archivo principal y, de ser necesario, con el filegroup secundario. Se realizaron consultas en las vistas del sistema (sys.databases) para confirmar la creación y se comprobó la correcta asignación de tamaños, crecimiento y rutas.
- **La Aplicación PHP Interactuara Correctamente:**
  Se testearon diversas ejecuciones desde el formulario web, evaluando tanto escenarios en los que se especificaban todos los parámetros como casos en los que se dejaban en blanco campos opcionales (por ejemplo, para no agregar filegroup secundario). La aplicación mostró mensajes claros sobre el éxito o fallo de cada operación.

- **Robustez y Seguridad:**
  Se validó que la generación dinámica de SQL fuera segura y que los errores se gestionaran adecuadamente sin exponer información sensible. Asimismo, se verificó que el uso de consultas parametrizadas evitara posibles ataques de inyección SQL.

**Dificultades Encontradas y Soluciones**

1. Errores de Sintaxis en SQL Dinámico

- **Dificultad:**
  Durante la construcción dinámica de la sentencia CREATE DATABASE, se presentaron errores de sintaxis, especialmente al concatenar los parámetros relacionados con el tamaño, crecimiento y la inclusión opcional del filegroup secundario.
- **Solución:**
  Se incorporó el uso de la función QUOTENAME() para proteger los nombres de objetos y se revisó cuidadosamente la concatenación de cadenas, asegurando la correcta inserción de espacios y comas. Además, se implementó una validación condicional para que el bloque de filegroup secundario solo se agregue si se proporcionan ambos parámetros (nombre y ruta), evitando bloques incompletos o redundantes.

2. Manejo de Parámetros Opcionales

- **Dificultad:**
  Era necesario permitir que ciertos parámetros (como los del filegroup secundario) fueran opcionales, sin que su ausencia generara errores en la sentencia final.
- **Solución:**
  Se implementaron condicionales en el stored procedure para evaluar si los parámetros opcionales eran distintos de NULL. Si estos parámetros no se proporcionaban, se omitía el bloque correspondiente en la construcción del SQL dinámico, lo que aseguraba que la sentencia final se formara correctamente.

3. Integración y Validación en la Aplicación PHP

- **Dificultad:**
  La integración del stored procedure en la aplicación PHP generaba algunos problemas en el manejo de errores y la interpretación de la respuesta, ya que la aplicación reportaba errores aunque la base de datos se creara correctamente.
- **Solución:**
  Se ajustó el manejo de la respuesta en PHP para que, en caso de que la consulta se ejecutara sin problemas (retornando el código de éxito), se mostrara un mensaje de confirmación personalizado (" Todo salió

correctamente."). Se utilizaron consultas parametrizadas a través de la extensión sqlsrv y se capturaron errores con sqlsrv_errors() para proporcionar retroalimentación clara sin generar mensajes erróneos en la interfaz.

4. Verificación y Depuración

- **Dificultad:**
  La depuración del SQL dinámico resultó complicada debido a que los mensajes de PRINT se generaban en el contexto del servidor, lo que dificultaba la correlación con los datos enviados desde la aplicación.
- **Solución:**
  Se agregaron sentencias PRINT @SQL; estratégicamente en el procedimiento, permitiendo visualizar la instrucción final generada. Esto facilitó copiar y pegar la consulta en SQL Server Management Studio (SSMS) para una depuración detallada y asegurar que todos los parámetros se concatenaban correctamente.

**Conclusiones**

- **Automatización Efectiva:**
  La implementación del procedimiento almacenado para la creación dinámica de bases de datos demostró ser una solución eficaz para automatizar un proceso tradicionalmente manual. La capacidad de configurar parámetros críticos, como rutas, tamaños, crecimientos y la opción de agregar filegroups secundarios ofrece una gran flexibilidad en la administración de recursos.
- **Robustez y Seguridad:**
  El uso de SQL dinámico, en combinación con funciones de seguridad como QUOTENAME(), y el manejo de errores mediante bloques TRY...CATCH garantizó que el sistema fuera robusto y seguro. Se minimizó el riesgo de inyección SQL y se logró capturar y reportar errores de manera clara.
- **Integración Exitosa con la Aplicación PHP:**
  La integración del stored procedure en la aplicación web, mediante el uso de la extensión sqlsrv para trabajar desde php con sql server, y un formulario diseñado con Bootstrap, permitió que los usuarios administraran las bases de datos de forma intuitiva y profesional. La interfaz no solo facilitó la entrada de datos, sino que también proporcionó retroalimentación clara sobre el éxito o fallo de cada operación.
- **Reducción de Errores y Ahorro de Tiempo:**
  Al automatizar la creación de bases de datos, se redujo significativamente la intervención manual, disminuyendo la probabilidad de errores humanos y optimizando el tiempo dedicado a tareas de administración.